SPACEBEL SAS

VTS

Equipe VTS

VTS-MU-G-69-SPB
Change : 05 Date : 06/03/2014
Issue : 01 Date : 14/11/2014

Ref. : VTS/MU/69

Distribution Code : E

USER MANUAL

VTS 2

Written by : Equipe VTS SPACEBEL SAS	Date : 14/11/2014	
Approved by :	Date :	
For application :	Date :	

Spacebel SAS, Technoparc 8, Rue Jean Bart, 31670 LABEGE (France), Tél. +33 5-32-09-09-09

INDEX SHEET

CONFIDENTIALITY : DLP	KEYWORDS : VTS, User Manual

TITLE :

User Manual
VTS 2

AUTHOR(S) :

Equipe VTS **SPACEBEL SAS**

SUMMARY : This document is the user manual of VTS.

RELATED DOCUMENTS : Stand alone document.	LOCALIZATION : VTS/MU

VOLUME : 1	TOTAL NUMBER OF PAGES : 185 INCLUDING PRELIMINARY PAGES : 0 NUMBER OF SUPPL. PAGES : 0	COMPOSITE DOCUMENT : N	LANGUAGE : EN

CONFIGURATION MANAGEMENT : NG	CM RESP. :

REASONS FOR EVOLUTION : Release of VTS 2.6 "Rosetta"

CONTRACT : Marché sous accord-cadre n°116754

HOST SYSTEM :

Microsoft Word 11.0 (11.0.5604)
C:\Documents and Settings\Administrateur\Application Data\Microsoft\Templates\Normal.dot
Version GDOC : v4.2.2.2
Base projet : \\wiki.spacebel.fr\kiwi\Outils\Gdoc\vts.mdb

INTERNAL DISTRIBUTION

Name	Entity	Internal Postal Box	Observations

EXTERNAL DISTRIBUTION

Name	Entity	Observations

CHANGES

Issue	Rev.	Date	Reference, Author(s), Reasons for evolution
05	01	14/11/2014	VTS/MU/69 Equipe VTS SPACEBEL SAS Release of VTS 2.6 "Rosetta"
05	00	06/03/2014	VTS/MU/69 Equipe VTS SPACEBEL SAS Release of VTS 2.5 English translation
04	04	03/12/2013	VTS/MU/69 Equipe VTS SPACEBEL SAS Livraison VTS 2.4 Fauchées senseurs Evènements Bugs fixes et fonctionnalités mineures
04	03	03/07/2013	VTS/MU/69 Equipe VTS SPACEBEL SAS Livraison VTS 2.3 Ergonomie de VTS Lancement dynamique des applications Liste des paramètres des applications Bugs fixes et fonctionnalités mineures
04	02	19/02/2013	VTS/MU/69 Equipe VTS SPACEBEL SAS Livraison VTS 2.2. Textures dynamiques Bugs fixes et fonctionnalités mineures
04	01	13/09/2012	VTS/MU/69 Equipe VTS SPACEBEL SAS Ajout de l'enregistrement des films.
04	00	24/07/2012	VTS/MU/69 Equipe VTS SPACEBEL SAS Livraison VTS 2.0
03	01	05/06/2012	VTS/MU/69 Equipe VTS SPACEBEL SAS Livraison VTS 1.3.1 (temps-réel)
03	00	27/02/2012	VTS/MU/69 Equipe VTS SPACEBEL SAS Livraison VTS 1.3.0

Issue	Rev.	Date	Reference, Author(s), Reasons for evolution
02	00	19/05/2011	VTS/MU/69 Equipe VTS SPACEBEL SAS Construction du manuel utilisateur depuis le contenu de la documentation Wiki
01	01	13/10/2010	VTS/MU/69 Equipe VTS SPACEBEL SAS Ajout du chapitre de présentation générale Complément des parties manquantes de la version 1.0 Chapitre sur les plugins
01	00	15/03/2010	CELEST/MU/69 Equipe VTS SPACEBEL SAS Création du document

TABLE OF CONTENTS

1. OVERVIEW

1.1. REFERENCE DOCUMENTS

These files are referenced in this user manual :

RD1 Définition du protocole d'échange CIC V1.0
LEGAL Jean-Luc, 09/06/2010, Issue 1, Rev. 0
DCT/DA /PA - 2009.0021267

1.2. APPLICABLE DOCUMENTS

There is no applicable document.

2. INTRODUCTION

2.1. OVERVIEW OF VTS

VTS is a visualization toolkit for space data. Its primary goal is to animate satellites in both 2D and 3D environments. Its architecture also allows it to be extended with any number of compatible applications.

The toolkit comes with a configuration utility, which allows the definition of the elements in the visualization: 3D models, geometry and mobile parts of the satellites, data sources for all positions, attitudes, angle of rotations, etc. It also allows configuration for satellite sensors and ground stations, for the visualization scenario, and for the applications involved in the visualization. VTS then uses this configuration to run the animation. It handles starting and synchronizing all the chosen applications. Its core element, the Broker, offers functionality for navigating in visualization time, controlling 3D cameras and defining a range of other display attributes. It also broadcasts visualization settings associated with each scenario state to client applications.

VTS is designed in a way that allows connected applications to control the visualization time (eg. time ticks from a simulator, time navigation in a plotting software, etc.). As for the visualization data, they can be provided either through files or network streams (which are broadcasted to all clients). The synchronization procotol for client applications is specific to VTS; however the data files are based on a CCSDS standard, allowing direct compatibility with all tools handling this european format.

On the technical side, VTS relies on Celestia as its default 3D visualization tool. Celestia is free software, well known for its performance and the realism of its rendering. All of Celestia's many features are thus available to VTS users. As for portability, the toolkit is written using C++ and the Qt framework, ensuring stable behavior across Linux and Windows platforms. Lastly, available client applications come from a variety of origins. As an example, the PrestoPlot plotting tool is fully interfaced with VTS.

With these characteristics, VTS becomes an essential tool for all data production activities in the satellites flight dynamics field. It enables graphical validation of behaviors and attitude strategies, and can be used as a practical exchange support for all actors in satellite AOCS and space mechanics.

2.2. VTS QUICK START

2.2.1. Installing the toolkit

The VTS toolkit comes as a zip archive. To install it, simply decompress it in a folder. This installation folder will be referred to as "VTS/" throughout this manual.

2.2.2. Starting the toolkit

The VTS toolkit can be started by double-clicking the "startVTS.exe" executable file under Windows, or by running the command "./startVTS" under Linux.

The main window that opens is the configuration utility. It is used to create a project and define its elements: satellites, sensors, ground stations, and client applications. This quick start guide will only cover opening and launching an existing project.

2.2.3. Opening an existing project

- Click the **Open project** button in the toolbar, or in the **File** menu.

- The **Open Project File** dialog opens. Select the *Cubesat.vts* project file located under *VTS/Data/CubeSat/* and click **Open**.
- The project is loaded into the project tree.

2.2.4. Starting the visualization

- To start the project visualization, click the **Run** button in the toolbar, or in the **Project** menu.
- The *Broker* window opens, with all the client applications defined for the project : the 2D window, Celestia, PrestoPlot, etc.

2.2.5. Interacting with the Broker

The Broker is the core application of the visualization phase. It starts the client applications, regulates the visualization time, sends user commands to the client applications, and much more.

2.2.5.1. Start of a visualization

When visualization is started from the VTS configuration utility, the Broker starts all the client applications defined in the VTS project. During this initialization phase, the Broker displays the Initializing message in the text fields of the time control area. Time does not start flowing until all client applications have signaled they are ready.

Upon connection of the various client applications, the Broker's tabs are populated with commands and information regarding these applications.

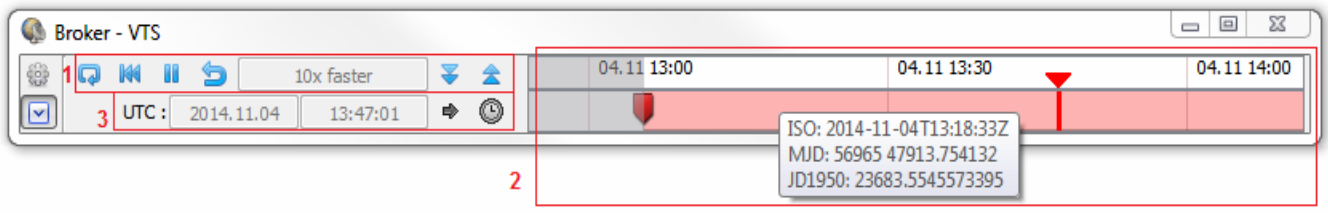
Visualization automatically starts to play with time ratio 1 once initialization is finished, unless specified otherwise in the VTS project.

2.2.5.2. End of a visualization

The visualization can be stopped directly by clicking the top-right cross button to close the Broker. It also stops once all client applications have been closed by the user, or if the VTS configuration utility is closed (if the visualization was started from there).

2.2.5.3. Time management

Controls and information regarding time are available in all Broker display modes. The following interface allows interacting with visualization time:



Time control areas

2.2.5.3.1. Time controls

The buttons in area (1) control time in all client applications:

- **Repeat**: Toggles loop playback for the visualization.
- **Restart**: Resets time to the project's start date. The play/pause status remains unchanged.
- **Play**/**Pause**: Plays/pauses the time flow. Pausing does not prevent interaction with the visualized entities in the client applications, or navigation in visualization time using the timeline.
- **Slower**: Decreases the time ratio. Two clicks on this button result in a 5 times slower time flow. The current time ratio is displayed next to this button.
- **Faster**: Increases the time ratio. Two clicks on this button result in a 5 times faster time flow.
- **Revert**: Every click on this button inverts the time flow direction. The time ratio remains unchanged.

2.2.5.3.2. Timeline

The timeline (2) displays the visualization's time range. Dragging the cursor controls the visualization's current time. While moving the cursor, the current time is kept updated in all client applications synchronously. For finer or coarser-grained time control, the timeline can be zoomed in/out using the mouse wheel.

2.2.5.3.3. Current time

A text area (3) displays the current visualization time. The default format is the ISO time format (date and time in UTC). The arrow button circles through other available time formats: CNES julian day (JD1950, fractional days, with reference date January 1st, 1950) and modified julian day (MJD, days and seconds, reference date November 17th, 1858). The Edit date... button pops up a dialog in which the current visualization time can be accurately defined (in all of the above time formats).

2.3. SOFTWARE REQUIREMENTS FOR VTS

This page describes the software requirements for proper operation of the VTS toolkit.

2.3.1. Write access permission

VTS needs write access permission in its whole folder hierarchy. Many tools create temporary files for configuration and visualization. Moreover, installation of new VTS clients consists in copying the client directory in the VTS/Apps folder.

2.3.2. Supported operating systems

OS	Version	Optimal for VTS	Tested	Comments
Windows	XP	X	X	SP3 recommended
	Seven	X	X	
	8.1		X	
Linux 32bits	RHEL 5/6 (KDE)	X	X	
	RHEL 5/6 (Gnome)		X	
	RHEL 3/4		X	Special build available on demand

OS	Version	Optimal for VTS	Tested	Comments
	Ubuntu		X	Additional packages may be required
	Debian 6/7		X	Additional packages may be required
Linux 64bits	RHEL 5/6	X	X	Additional packages may be required for RHEL 5
	Ubuntu		X	Additional 32bits compatibility packages are required
	Debian 6/7		X	Additional 32bits compatibility packages are required
Mac OS	X 10.7 *Lion*			Available soon

2.4. HARDWARE REQUIREMENTS FOR VTS

This page describes the hardware requirements for proper operation of the VTS toolkit.

2.4.1. Recommended system requirements

- Core™ i5 processor or better
- 4 GB RAM memory
- Video card (nVidia is recommended for Celestia) with 1 GB memory
- 1 GB free disk space

For nVidia video cards, prefer midrange cards : GeForce [23456]50 series.

Here are some example systems on which VTS would run very smoothly:

	Laptop	Desktop
Processor	3rd gen Intel® Core™ i7-3610M (3.30GHz, 6MB)	3rd gen Intel® Core™ i7-3770 (3.40GHz, 8MB)
Memory	Dual-channel SDRAM DDR3 4 GB, 1600 MHz	Dual-channel SDRAM DDR3 8 GB, 1600 MHz
Video	NVIDIA® GeForce® GT 650M 2 GB	NVIDIA® GeForce® GT 640 1 GB
Hard drive	SATA 1TB 5400 rd/min	SATA 1TB 7200 rd/min

2.4.2. Minimal system requirements

- 1GHz processor
- Dedicated video card with 128 MB memory (nVidia GeForce 5000 or equivalent), embedded video card for modern processors (Intel HD Graphics 3000)
- 512 MB memory
- 500 MB free disk space

Such a system will run VTS, but won't handle the more realistic rendering options in Celestia.

2.4.3. Video card drivers

In order to get the best performance from a machine, video card drivers should be kept up-to-date. This will solve the majority of display problems encountered while using Celestia.

Driver updates are available on the website of the card's chipset maker (nVidia, ATI or Intel), or on the website of the system maker for some laptops.

Issues with Celestia and sensor swath: if Celestia closes forcefully while running a visualization with sensor swath enabled, it may be due to outdated graphics card drivers. If drivers are up-to-date and the problem persists, with an Intel chipset, it may be because the chipset is incompatible with the rendering of sensor swath items. Sensor swath must then be disabled for the project (see http://www.opengl.org/wiki/FAQ#Why_is_my_GL_version_only_1.4_or_lower.3F).

3. GENERAL CONCEPTS

3.1. CONVENTIONS IN VTS

As in every software, VTS sets up conventions for some of its elements. This page lists those conventions.

3.1.1. Frames

In VTS, frame axes form an orthonormal direct basis.

3.1.2. Quaternions

The first element of a quaternion corresponds to its scalar part.

3.2. DATE FORMATS IN VTS

3.2.1. Formats list

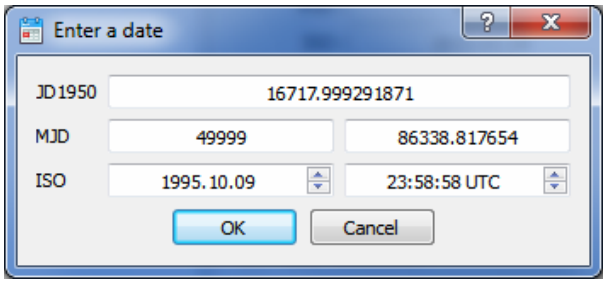
Date format	Abbreviation	Example: May 17th, 2011, 16h22	Distance to JD1950	Leap seconds	Usage in VTS
CNES Julian day	JD1950	22416.6819444444	0	Yes	• Synchronization protocol • Display • Project file
Modified julian day	MJD	55698 58920.000000	33282	Yes	• CIC/CCSDS files • Display • Project file
Calendar date	ISO	2011.05.17 16:22:00	NA	Yes	• Display
ISO 8601 date (forced UTC)	ISO	2011-05-17T16:22:00.000Z	NA	Yes	• Project file
Celestia Julian day	jjCEL	2455699.181944	2433282.5	No	• Celestia data files

The internal date representation in VTS uses the reference date of JD1950, and is expressed with two fields for days and seconds (like MJD).

VTS uses the time system UTC both for display and internal calculations.

3.2.2. Entering a date

When a date needs to be specified anywhere in VTS, the same user interface is used. The date can be entered in the JD1950, MJD and ISO formats.



Date dialog in VTS

3.3. DATA SOURCES IN VTS

There are three types of data sources in VTS:

- *Fixed*: a fixed value
- *File*: a time-dependent value read from a file
- *Stream*: a value provided by a network stream

3.3.1. Fixed data source

This data source provides a constant, user-defined value. The actual value is defined by the user upon selection of the source.

This source can be used eg. for the position and orientation of fixed satellite components and sensors.

3.3.2. File data source

This data source is based on a file, specified by the user upon selection of the source. The values are read from the file, which must follow the standard CIC/CCSDS format.

This source can be used for the ephemerides of celestial bodies, satellites, as well as for the position and orientation of mobile parts on satellites. It is most adequate for offline visualization, once the position and attitude data have been saved to files.

3.3.3. Stream data source

This data source is based on a network stream, defined by the user upon selection of the source.

This source can be used for the ephemerides of celestial bodies, satellites, as well as for the position and orientation of mobile parts on satellites. It is most adequate for online visualization, in which the position and attitude data are streamed on-the-fly by external applications, such as a simulator.

The Stream data source provides two modes of operation:

- *INTERPOL*: in order for visualization clients (2dWin, Celestia, etc.) to use interpolated values, the Broker will introduce a slight delay between data transmission and time synchronization with visualization clients.
- *DIRECT*: the data update rate is not taken into account while computing time synchronization delay. Beware that no extrapolation is done in standard VTS clients.

See also the documentation for Real-time VTS in the Synchronization protocol for VTS clients chapter.

3.4. POSITION OF OBJECTS IN VTS

The position of an object in VTS is defined as a translation from the object's reference frame towards its location. The position can be fixed, sampled, or streamed. This section is dedicated to the contents of position data, no matter its source. For a definition of the various data sources, see the Data sources in VTS section.

3.4.1. Reference frames

Each object in VTS has its position defined in a reference frame. For example, the position of solar arrays is relative to the satellite frame.

An object is located at the origin of its reference frame when its position vector is null.

Reference frames for the positions of various object types are defined in the table below:

Object type	Reference frame
Central body	Parent object's frame
Ground station	Central body's local frame
Satellite	EME2000, Earth Mean Equator at epoch J2000
Component	Satellite's frame for a top-level component Parent component's frame for components of other levels
Satellite sensor	Satellite's frame for a top-level sensor Parent component's rotation center for sensors attached to a component
Ground station sensor	Ground station's frame

3.4.2. Position of satellite components

The position for a top-level component is defined in the satellite's frame. The position for a sub-component, i.e. a component not directly attached to a satellite but rather to another component, is defined in its parent component's frame.

When a component is translated or rotated, all its sub-components are translated and rotated as well. This can be illustrated by the operation of a robotic arm: each segment is animated locally relative to the previous segment.

There is a special case to the relative position of sub-components, when using 3D files containing the position of sub-components relative to the satellite's frame. This case is discussed in section 3D file format in VTS.

3.4.3. See also

- Conventions in VTS
- 3D file format in VTS
- Examples for position and orientation of objects in VTS

3.5. ORIENTATION OF OBJECTS IN VTS

The orientation of an object in VTS is defined as a rotation from the object's reference frame towards its attitude. The orientation can be fixed, sampled, or streamed. This section is dedicated to the contents of orientation data, no

matter its source. For a definition of the various data sources, see the Data sources in VTS section.

3.5.1. Reference frames

Each object in VTS has its orientation defined in a reference frame. For example, satellites around Earth use reference frame EME2000.

An object is aligned with the axes of its reference frame when its rotation vector is null. In the case of a satellite defined by a 3D file (see the 3D file format in VTS section), the X axis of the 3D mesh is aligned with the X axis of the EME2000 frame. The same applies for the Y and Z axes.

Reference frames for the orientations of various object types are defined in the table below:

Object type	Reference frame
Central body	Parent object's frame
Ground station	-Z : body center +X : towards geographic North pole At poles : undefined
Ground station sensor	Ground station's frame
Satellite	EME2000, Earth Mean Equator at epoch J2000
Component	Satellite's frame for a top-level component Parent component's frame for components of other levels
Satellite sensor	Satellite's frame for a top-level sensor, component frame for sub-level sensors. Aligned with the satellite's frame and the sensor points along the Z axis. Parent component's frame for a sensor attached to a component

3.5.2. Rotation center

The rotation center defines the point around which a 3D object is rotated. It depends on the object type, and is defined in the table below:

Object type	Rotation center	Reference frame for the rotation center
Central body	Central body's center	Central body's frame
Ground station	Ground station's position	Central body's frame
Ground station sensor	Sensor's position	Ground station's frame
Satellite	Center of gravity (user-defined)	Satellite's frame
Component	Rotation center (user-defined)	Component's frame
Satellite sensor	Sensor's position	Satellite's frame for a top-level sensor Parent component's frame for a sensor attached to a component

3.5.3. Orientation types

In VTS, an orientation can be defined by several types of rotations in the reference frame.

3.5.3.1. Quaternion

A quaternion defines a rotation from an object's reference frame towards its local frame.

The convention used for a quaternion sets its real part as the first component.

A null rotation is defined by the following quaternion:

```
1.0 0.0 0.0 0.0
```

All orientations can be reached. Quaternions are normalized by VTS. An all-zeroes value (0 0 0 0) is invalid.

3.5.3.2. Euler angles

Euler angles define a sequence of three rotations from an object's reference frame towards its local frame.

The convention used for Euler angles defines the order of rotations Z, X', Z'' as:

- Precession around axis Z of the reference frame
- Nutation around axis X' of the frame resulting from the precession
- Intrinsic rotation around axis Z'' of the frame resulting from the two previous rotations

Angles are expressed in degrees.

All orientations can be reached. There is no invalid value.

3.5.3.3. Axis and angle

Axis and angle rotations allow easy definition of the orientation of objects physically rotating around an axis, e.g. solar arrays.

The rotation axis must be defined, i.e. non null (0 0 0). Its coordinates are expressed in the object's reference frame.

The rotation angle in degrees is applied to the object around the rotation axis, in the counterclockwise direction. A positive angle implies a counterclockwise rotation around the axis ; convention dictates that this corresponds to screwing along the direction of the axis, i.e. in clockwise direction while looking along the direction of the axis.

3.5.3.4. Direction

A direction defines a rotation so that the X axis of the object points in the specified direction. It should be noted that this kind of orientation leaves a degree of freedom (3rd rotation of the Euler angles), and should thus only be used for solids of revolution around the X axis (e.g. a vector along the X axis).

3.5.3.5. Azimuth and elevation

Azimuth and elevation define a sequence of two rotations, first around axis Z then around axis -Y' (resulting of the previous rotation). If both rotations are null, the object points towards X. As for the direction orientation, this kind of orientation leaves a degree of freedom around the aim axis. Notice that in this orientation mode, conventions for sensor orientations pointing toward +Z is changed for pointing towards +X.

3.5.4. Validity domains

The different kinds of orientations are not relevant for all object types. The table below lists the valid orientation kinds for all object types. It should be noted that combinations marked as not relevant are still available in VTS, so that they may be used should the need arise. Combinations marked with a dash are however not possible to use with VTS.

Object type	Quaternion	Euler angles	Axis and angle	Direction	Azimuth elevation
Central body	**Yes** *E.g.: Comet*	Possible	No	No	No
Satellite	**Yes** *E.g.: Attitude*	Possible	No	No	No
Component	**Yes** *E.g.: Mobile part*	**Yes** *E.g.: Onboard instrument*	**Yes** *E.g.: Solar arrays*	**Yes** *E.g.: Representation of a vector*	**Yes** *E.g.: Direction of an antenna*
Satellite sensor	**Yes** *Ex.: Star tracker*	**Yes** *Ex.: Star tracker*	No	**Yes**	**Yes**

NB: Currently, the orientation of ground stations and ground station sensors cannot be modified.

3.5.5. See also

- Examples for position and orientation of objects in VTS
- Conventions in VTS
- 3D file format in VTS

3.6. EXAMPLES FOR POSITION AND ORIENTATION OF OBJECTS IN VTS

This section illustrates the positioning and orientation of an object with regards to its parent satellite. Relevant definitions can be found in the Position of objects in VTS and Orientation of objects in VTS sections.

3.6.1. Methodology

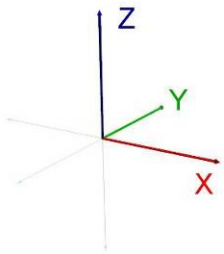
In order to fully understand the mechanisms at hand in the 3D views of VTS for the display of satellites and their parts (components, sensors, etc.), the following parameters will be analyzed:

- Position relative to the satellite (or in a more general way, relative to the parent object)
- Orientation, with the following parameters:
  - Center of rotation
  - Axis of rotation
  - Angle of rotation

The position and orientation (be it quaternion, Euler angles, axis and angle, etc.) of an object can be defined in the Position/orientation dialog of the object. For components, the center of rotation can be defined in the 3D Properties dialog. Refer to the VTS configuration utility user manual for more details.

The 3D object used for this demonstration is the 3-axes trihedral usually used in VTS to materialize reference frames. The X, Y, and Z axes are respectively colored in red, green, and blue. This object represents here the satellite or any parent object. The length of each axis is one meter. Then, the same model is used, scaled down and

tarnished, to represent the object positioned by the transformations described below.

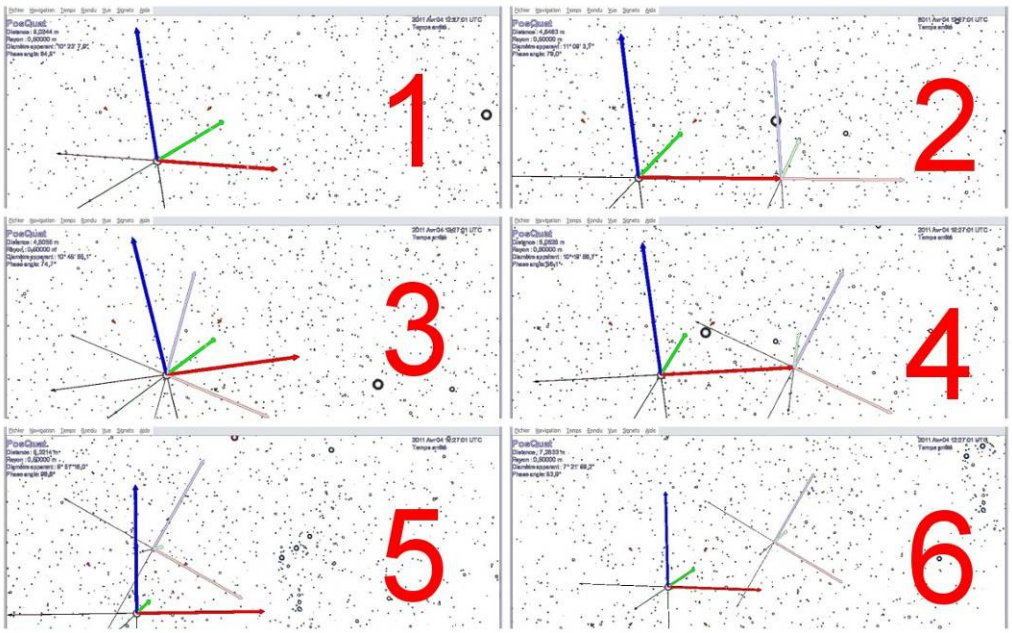


Materialization of the axes of the reference frame

3.6.2. List of tests

Number	Designation	Position	Center of rotation	Axis of rotation	Angle of rotation
1	No modification	X=0 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	0°
2	Single translation	X=1 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	0°
3	Single rotation	X=0 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	30°
4	Translation and rotation	X=1 Y=0 Z=0	X=0 Y=0 Z=0	X=0 Y=1 Z=0	30°
5	Offset rotation	X=0 Y=0 Z=0	X=1 Y=0 Z=0	X=0 Y=1 Z=0	30°
6	Translation and offset rotation	X=1 Y=0 Z=0	X=1 Y=0 Z=0	X=0 Y=1 Z=0	30°

3.6.3. Illustration of the result



Examples of various positions and orientations

3.6.4. Description of the result

3.6.4.1. Case 1: no modification

- Position: X=0 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 0°

The position and the angle of rotation are null. In this case, the object is at the same position and is aligned with its reference frame. It should be noted that since the angle of rotation is null, the axis and center of rotation have no influence.

3.6.4.2. Case 2: single translation

- Position: X=1 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 0°

In this case, the position of the object has been modified. Its value defines a position 1 meter forward along the X axis. The object is now centered at the end of the X axis of the reference frame.

3.6.4.3. Case 3: single rotation

- Position: X=0 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

The object's position vector is null and its center is located at the origin of its reference frame.

A single rotation has been applied. The center of rotation is null. The axis of rotation is 0 1 0, i.e. the rotation is counterclockwise around the Y axis (as if screwing in the direction of the axis). The diagram shows that the Y axis is shared by the two objects, while the X and Z axes have rotated 30°. The direction of the rotation can also be observed, clockwise when looking in the direction of the axis of rotation.

3.6.4.4. Case 4: translation and rotation

- Position: X=1 Y=0 Z=0
- Center of rotation: X=0 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

This case combines both cases. It illustrates the order in which the transformations are applied: the object is first rotated, then translated to its position.

3.6.4.5. Case 5: offset rotation

- Position: X=0 Y=0 Z=0
- Center of rotation: X=1 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

For a single rotation, the origin of the reference frame is used as rotation center. The rotation center can be modified so that the object is rotated around another point (e.g. for solar arrays). The coordinates of the center of rotation are 1 0 0. The object is rotated 30° around an axis located at the end of the red X axis, and directed along the green Y axis. Note that this rotation axis is not displayed on the diagram, hence the object appearing as if translated and not only rotated, due to the rotation center being offset.

3.6.4.6. Case 6: translation and offset rotation

- Position: X=1 Y=0 Z=0
- Center of rotation: X=1 Y=0 Z=0
- Axis of rotation: X=0 Y=1 Z=0
- Angle of rotation: 30°

This example adds a translation of 1 meter along the X axis on top of the transformations of case 5. The diagram confirms that rotation occurs before translation.

3.7. BACKWARDS COMPATIBILITY IN VTS

3.7.1. Backwards compatibility of project files

3.7.1.1. Principle

VTS features backwards compatibility of its project files. This means that project files from an earlier version of VTS will still be read correctly by newer versions of the toolkit. However, they will always be saved in the most recent file format if they are modified.

This mechanism is available from VTS 1.1 onwards.

3.7.1.2. Compatibility table

From revision	Toolkit version	Revision tag
From r3829 onwards	2.6	<project>

From revision	Toolkit version	Revision tag
From r3244 onwards	2.4, 2.5	<project>
From r1991 to r3243	2.0, 2.1, 2.1.1, 2.2, 2.3	<project>
From r1697 to r1990	1.3.1	<project>
From r1554 to r1696	1.3.0	<project>
From r1246 to r1553	1.2.3	<project>
From r1050 to r1245	1.2.1	<general>
From r1000 to r1049	1.1	<general>
Before r1000	NA	NA

3.7.2. Upgrading VTS

When a new version of VTS is available and if VTS can reach the http://msweb.cst.cnes.fr:9350/animgraph/vts/updater/news.htm webpage (i.e. on the internal network at CNES), a popup message is displayed to indicate the availability of a new VTS version. This message only appears when using the VTS configuration utility.

3.8. OBJECT PATHS IN VTS

The various objects displayed by VTS can all be addressed according to a naming scheme. For practical reasons, this naming scheme is directly derived from the one used by Celestia.

3.8.1. Definitions

The table below lists all the definitions used to identify objects in VTS:

Identifier	Description	Examples
`name`	Name of the object	Earth GS
`fullName`	**Full identifier of object**	Sol/Earth Sol/Earth/CubeSat/GS
`parentPath`	Identifier of the object's parent	Sol Sol/Earth/CubeSat
`celestiaRefName`	Name of the object's reference in Celestia	Earth GS_ref
`celestiaParentFullName`	Path to the object's parent in Celestia	Sol Sol/Earth/CubeSat_ref/CubeSat
`celestiaFullRefName`	Full path to the object's reference in Celestia	Sol/Earth Sol/Earth/CubeSat_ref/CubeSat/GS_ref

Identifier	Description	Examples
`celestiaFullName`	Full path to the object in Celestia	Sol/Earth Sol/Earth/CubeSat_ref/CubeSat/GS_ref/GS

3.8.2. Central body of an object

Objects orbiting Earth are identified starting with the full path to Earth, i.e. Sol/Earth.

3.8.3. Usage

The object name (name) and the path to its parent (parentPath) are the two main elements visible to VTS users. They can be found most notably in:

- The VTS project file format
- The Synchronization protocol for VTS clients

3.9. CENTRAL BODIES IN VTS

3.9.1. Categories of bodies

Central bodies in VTS a celestial objects around which orbit the main objects of a visualization, i.e. satellites.

By default, VTS supports the following list of central bodies in its projects:

- Mercury
- Venus
- Earth
- Mars
- Jupiter
- Saturn
- Uranus
- Neptune

Other bodies may also be partially supported: bodies such as the Moon, or some moons of solar system planets, are supported out of the box by Celestia. Hence they already come with predefined textures, but without ephemerides (in client applications other than Celestia).

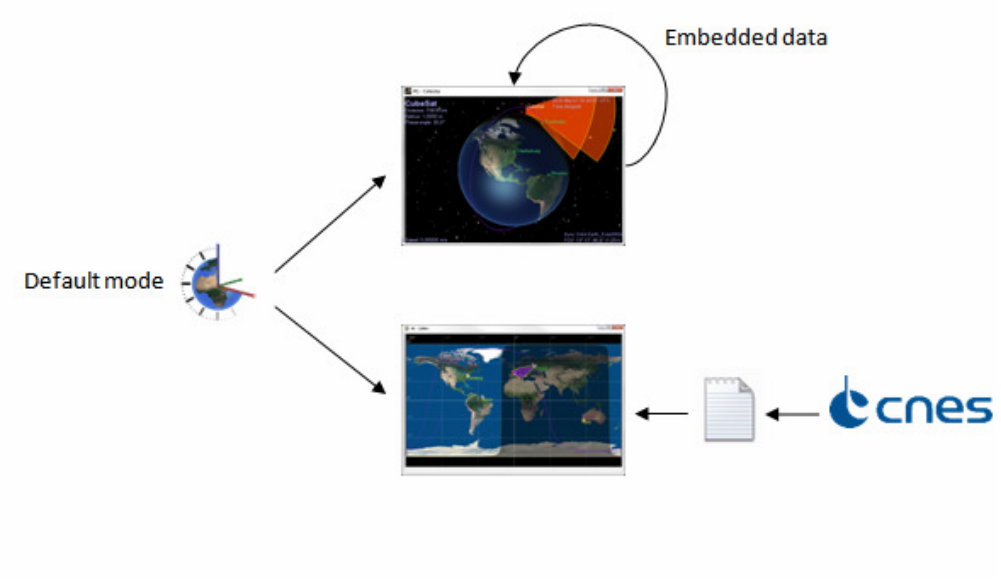
Custom central bodies can be added to a project. However, their appearance and ephemerides must be fully configured by the user.

3.9.2. Origin of ephemeris

3.9.2.1. Default ephemeris mode

Client applications choose their default mode to select the central bodies ephemerides. They can use their own calculation or shipped data, or VTS ephemerides catalog. This option can be also used for application that don't animate the central body or if this information is not relevant for them.

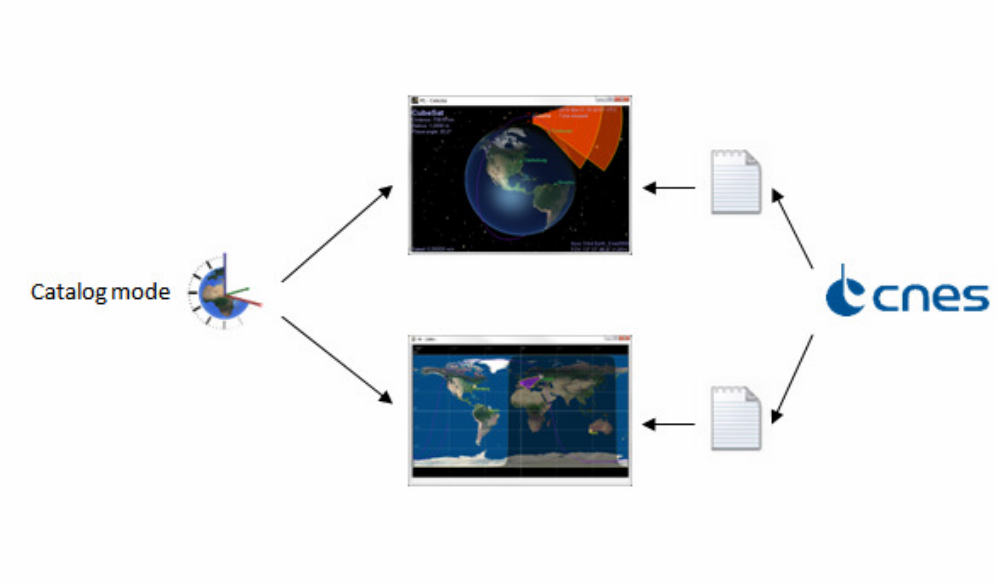
In this mode Celestia uses its own calculation, and 2DWin uses catalog ephemerides files.



Ephemeris source in default mode

3.9.2.2. Catalog ephemeris mode

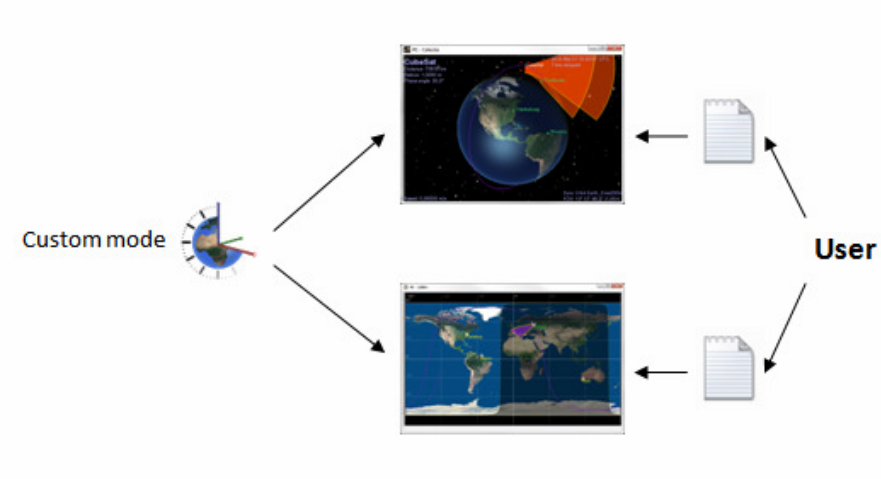
Client applications use the VTS ephemeris catalog data in this mode. The ephemeris catalog contains CIC files describing the position and the orientation of the solar system planets between 01/01/1950 and 31/12/2100. Position files contain one point per day and orientation files contain 6 point per day. These files are provided by CNES.



Catalog ephemeris mode

3.9.2.3. Custom ephemeris mode

Redefine the ephemerides of an existing Celestia body or define the ephemerides of a body not available in Celestia. Custom ephemerides must be defined if the current body is not available in Celestia.



Custom ephemeris mode

3.9.3. Central body frame

Coordinates of a central body are expressed in the heliocentric EME2000 frame.

North Pole of a central body is always the rotation North. It is used to display body axes and the planetographic grid.

3.10. STRUCTURE OF SATELLITES IN VTS

The structure of satellites in VTS matches the organization of the elements composing a satellite.

This structure can be found in:

- the VTS configuration utility, in the tree hierarchy displayed in the left pane. Refer to the VTS configuration utility user manual chapter.
- the contents of the project file. Refer to the VTS project file format chapter.
- the *View Properties* and *3D Cameras* tabs of the Broker. Refer to the Broker user manual chapter.

The structure can be described as follows:

- A project contains central bodies and satellites
    - A satellite is composed of a main component which defines the body of the satellite.
        - A component may contain sub-components.
            - These may themselves contain sub-components.
        - A component may contain sensors.

The naming scheme of objects relies on this structure, e.g. Sol/Earth/CubeSat/GS. See also the chapter about Object paths in VTS.

NB: Throughout this documentation and the VTS GUI, satellite components are sometimes referred to as "subparts".

3.11. SENSORS IN VTS

A sensor is either an onboard element on a satellite or an element of a ground station. This section presents the properties of sensors in VTS, and their representation in client applications.

3.11.1. Properties of a sensor

When orientated by a quaternion, euler angles or axis and angles, a sensor is defined by a conical aim volume around the Z axis, its apex being located at the origin of the sensor's local frame.

The sensors's base can be either elliptical of rectangular.

The cone is characterized by two half-angles around the X and Y axes (see the Orientation of objects in VTS chapter concerning the reference frame). These angles define rotations of the sensors's aim axis around the X axis (in the YZ plane) and Y axis (in the XZ plane).

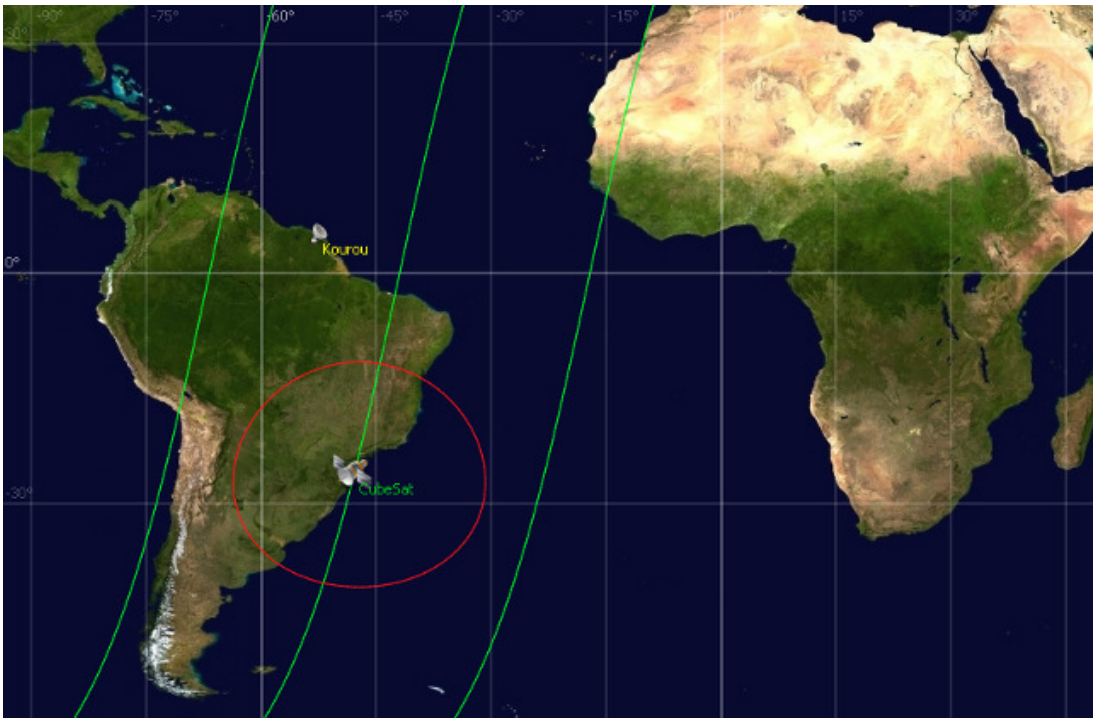
When orientated by a direction or altitude and azimuth, an additional rotation is made: X becomes the main axis, i.e. the sensor looks at the given direction. Since the third (self) rotation is not reliable, this orientation mode should be used only with elliptical sensors with similar X and Y apertures (circular sensors).

There is currently no limit to the effective range of a sensor.

3.11.2. Visualization of a sensor

3.11.2.1. Satellite sensor

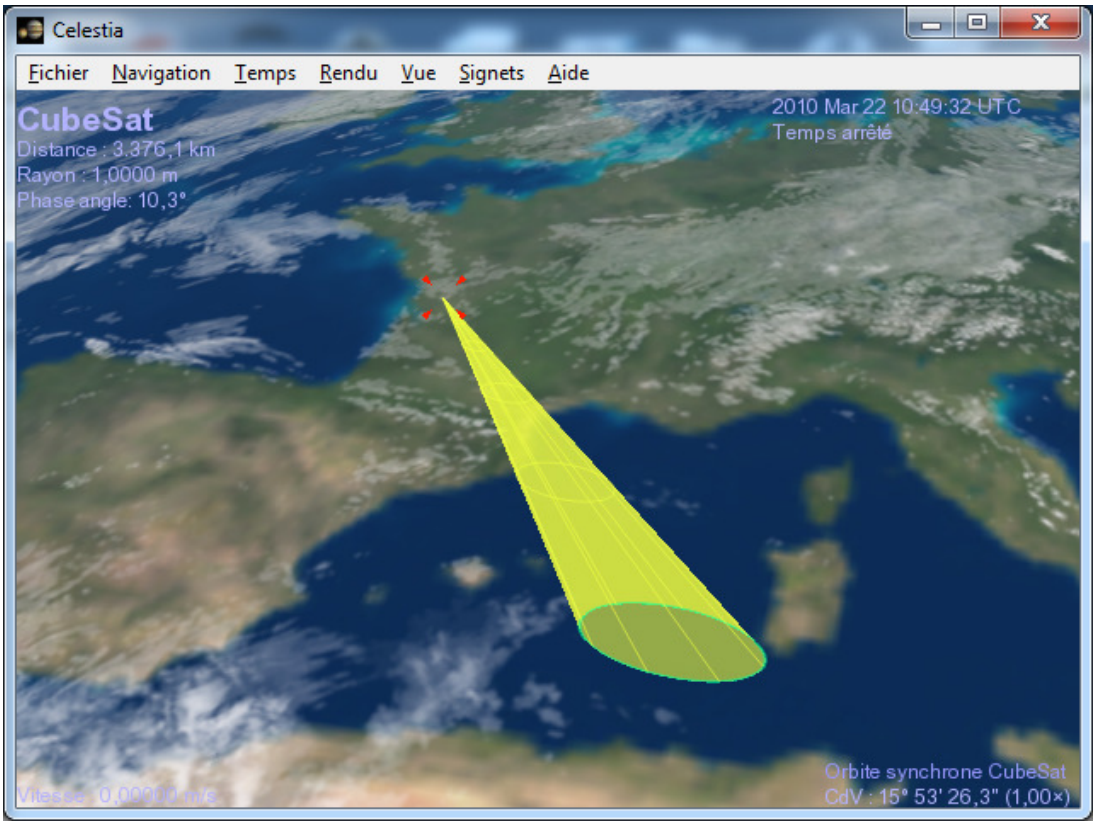
In the 2D view, a satellite sensor is represented by the intersection of its volume with its target central body.



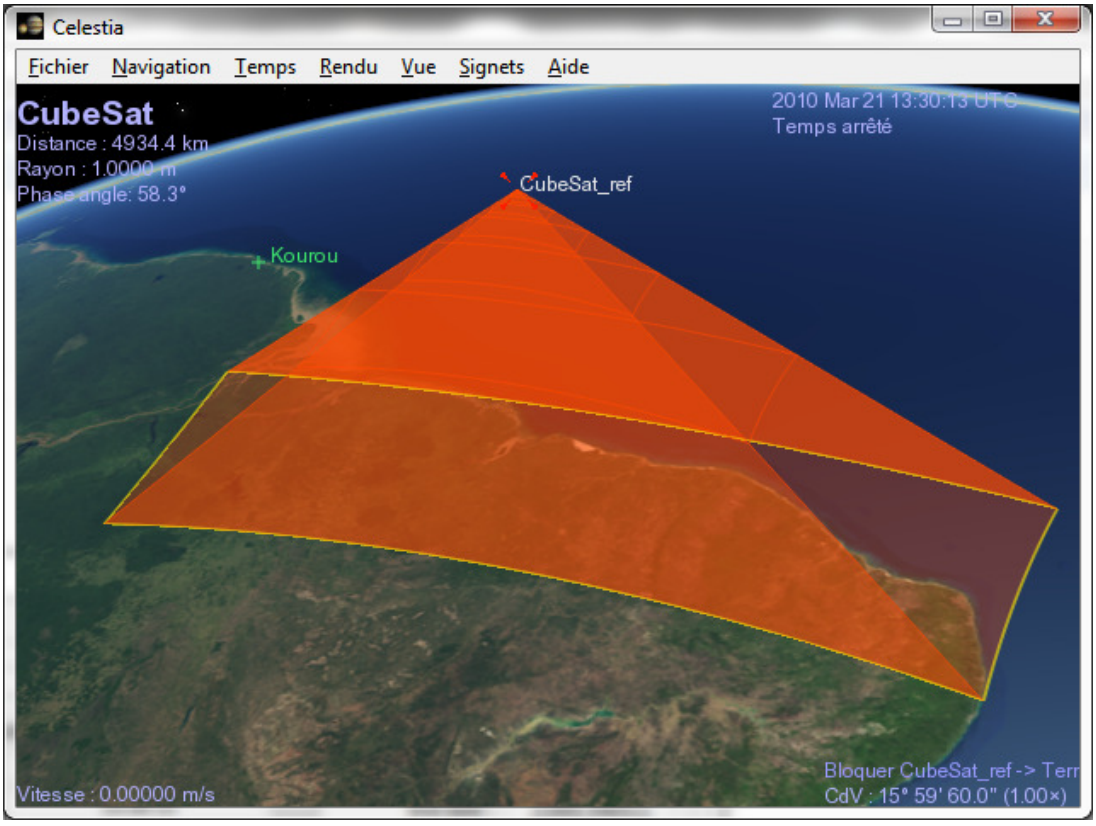
Elliptical sensor in 2DWin

Rectangular sensor in 2DWin

In the 3D view, a satellite sensor is represented by its aim volume. In order not to overload the visualization, a maximum range is defined for the sensor in the VTS configuration utility. Beyond this range, the sensor's volume will not be displayed. As a rule of thumb, the maximum sensor range should be taken equal to the maximum distance between the satellite and the center of the central body. Beware that a too short range could result in an erroneous computation of the sensor volume's intersection with the sensor target.

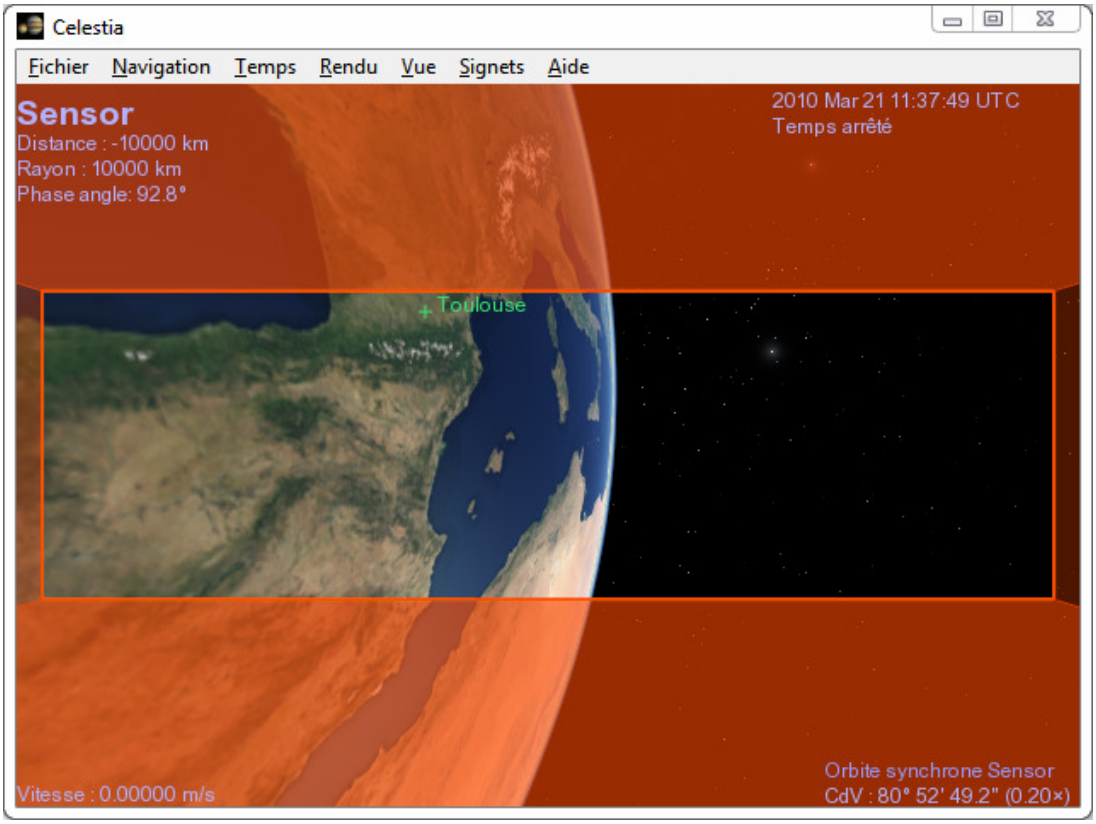


Elliptical sensor in Celestia



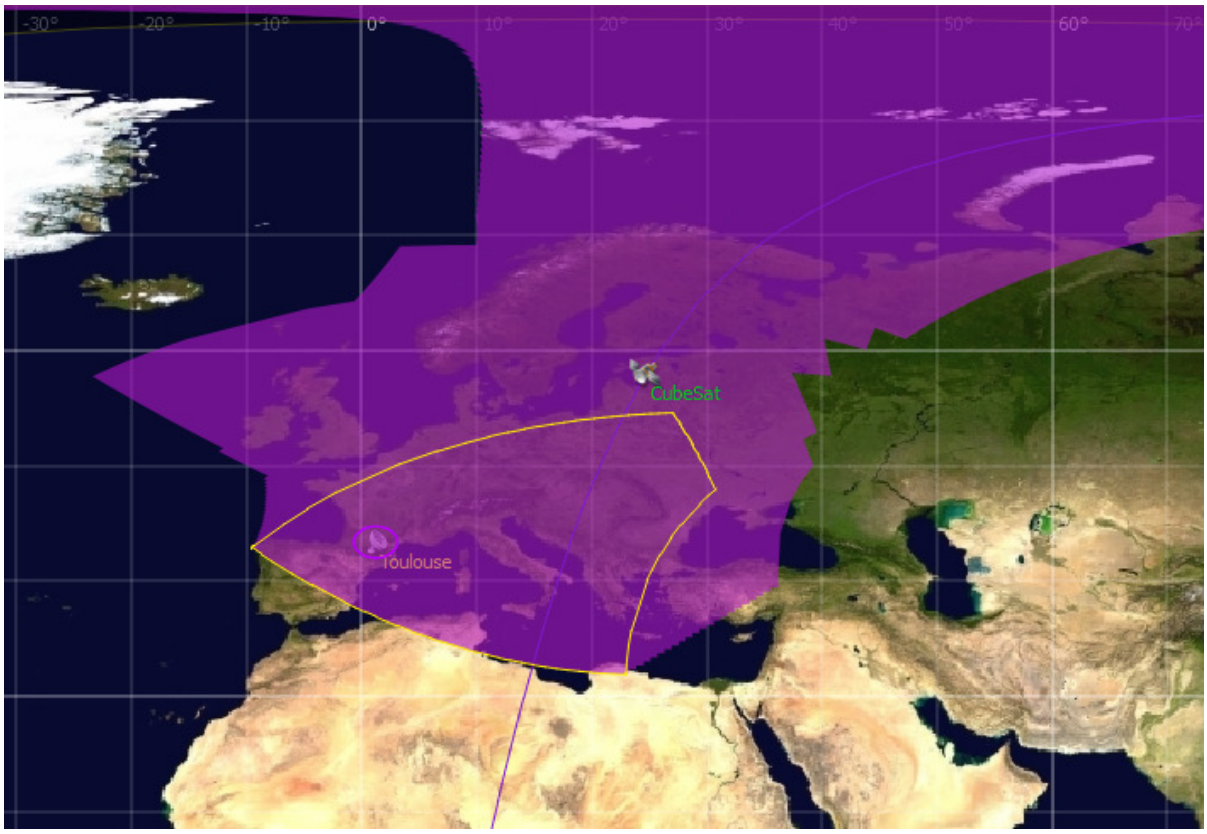
Rectangular sensor in Celestia

Moreover, the 3D view allows rendering the scene from the point of view of the sensor, so that what the sensor perceives can be displayed (refer to the 3D Cameras tab section in the Broker's user manual).

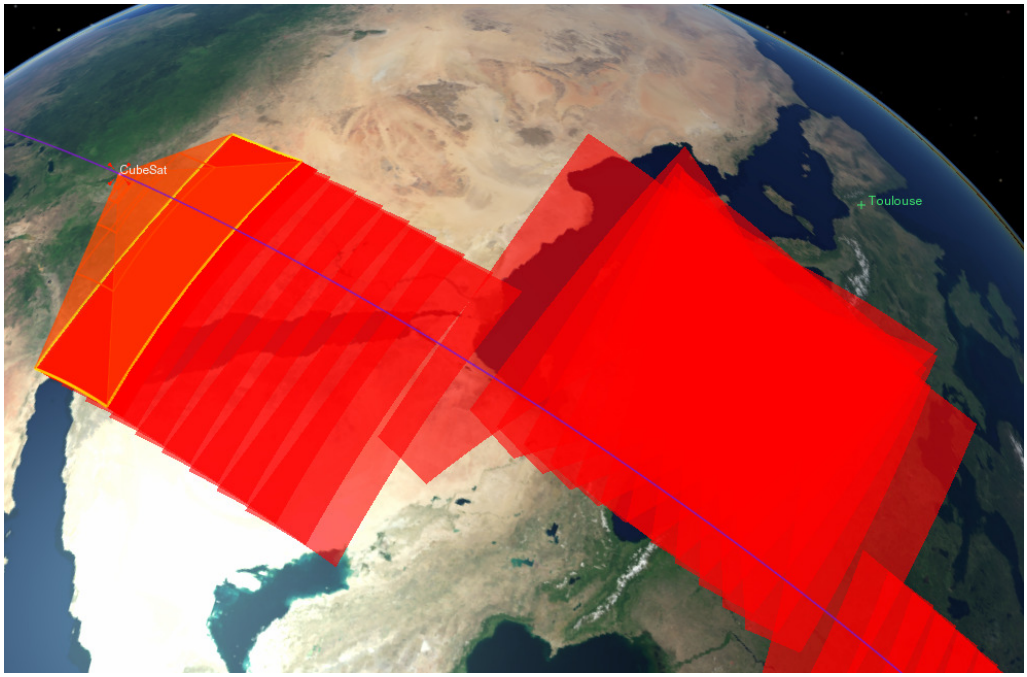


Sensor camera in Celestia

VTS also allows displaying the sensor's swath, both in 2DWin and Celestia.



Sensor swath in 2DWin



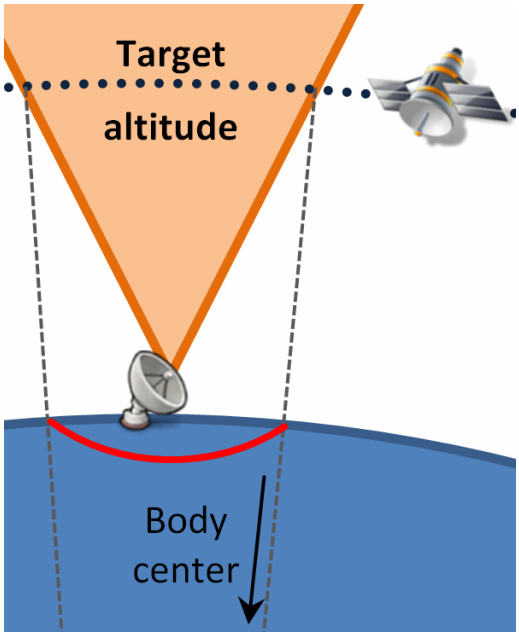
Sensor swath in Celestia

Caution: To preserve reasonable performances on the widest possible range of computers, the computation algorithm for sensor swath in Celestia is approximate. This has the consequence that when a sensor covers a large portion of its target body, the computed sensor swath is erroneous near the sides of the body disc.

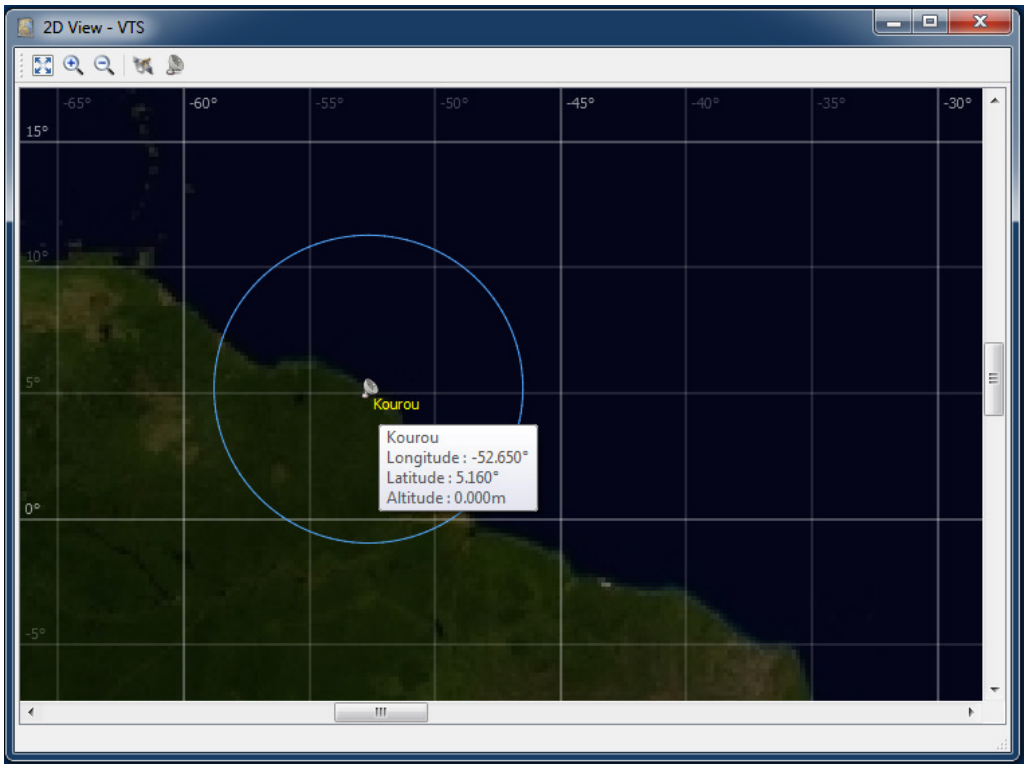
In case of forceful close of Celestia when visualizing sensor swath: see the section on Hardware requirements for VTS.

3.11.2.2. Ground station sensor

In the 2D view, a ground station sensor is represented by the projection of the base of its conical aim volume, at a given altitude, back on the surface of its central body. The altitude can also be set to the altitude of one of the project's satellites, and will then vary with time. A ground station sensor has the same parameters as a satellite sensor, with its aim axis directed right above its ground station. The angles define the half-angles of the sensor's cone around the X and Y axes. In order to obtain the visibility circle at minimum elevation, the half-angles should be set to the complementary angles of the half-angles (90° - half-angle around X or Y).



Projection of a ground station sensor in 2DWin



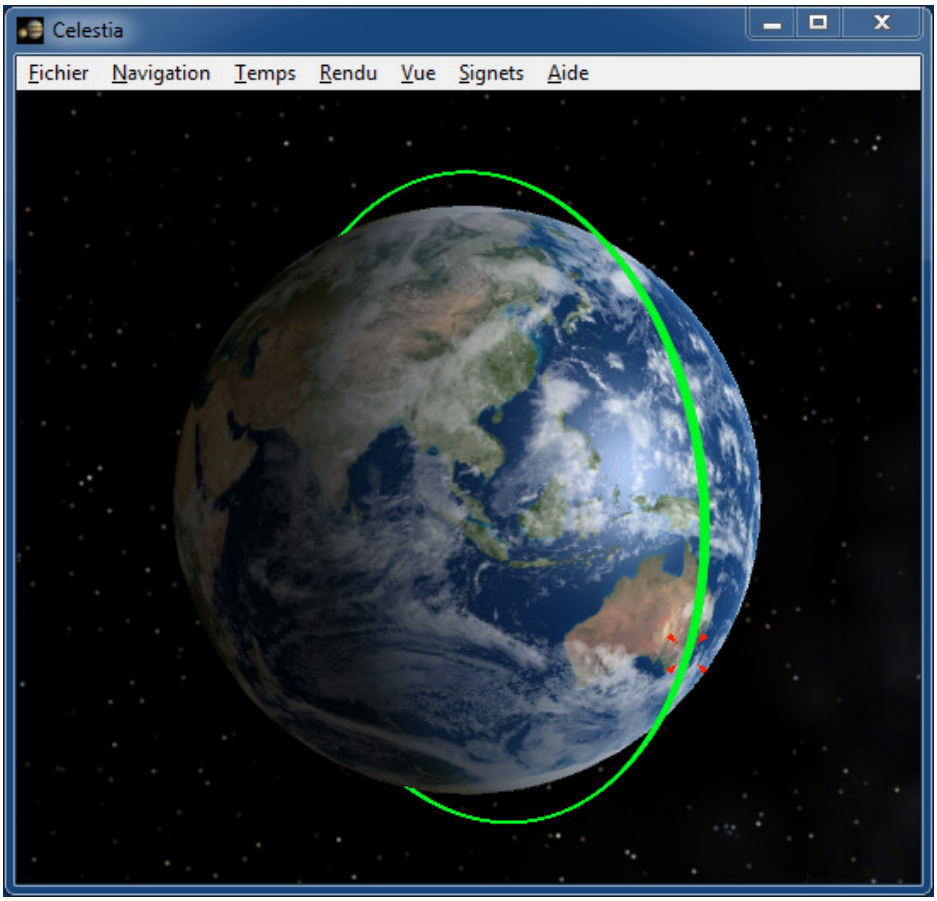
Ground station sensor in 2DWin

Ground station sensors are not displayed in Celestia.

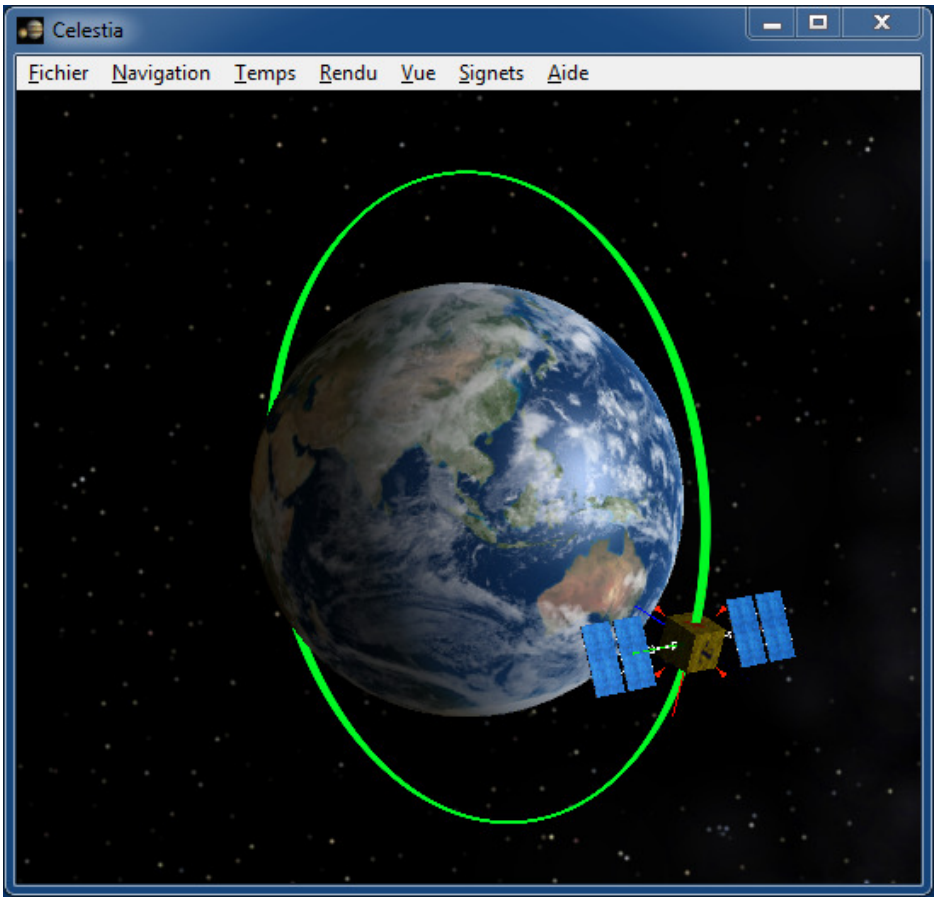
3.12. SCALE FACTORS IN VTS

Scale factors in VTS can be applied to central bodies or satellites of a project, for Celestia and client applications that support scale factors. This feature allows easy visualization of both the satellite's attitude and its orbital position. All satellite components are affected by the satellite's scale factor. Note that when a central body's scale factor is different than 1, the computations for sensor-body intersection will be erroneous in clients affected by the scale factor.

The scale factor for central bodies and satellites can be dynamically adjusted in the View Properties tab of the Broker, using the Body scale and Satellite scale properties.

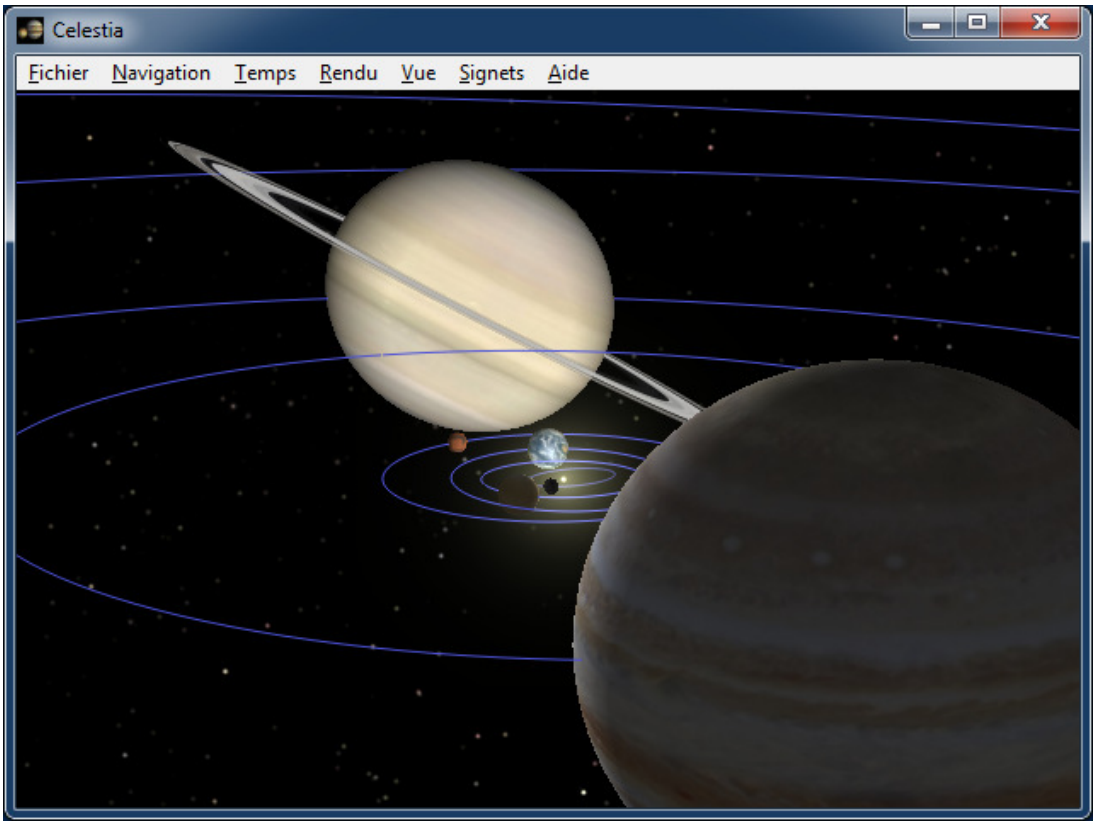


Scale factor 1 for Earth and the project's satellite



Scale factor 0.75 for Earth and 500 000 for the project's satellite

Scale factors also allow for easy visualization of motion for the whole solar system. The Solar system scale property in the View Properties tab of the Broker allows dynamically setting the scale factor of the whole solar system. Note that the Sun is not affected by this scale factor. Scale factors can be set for the whole solar system and for independent central bodies within in, e.g. to decrease the size of the gas giants.



Scaled solar system

The scale factor editor in the view properties editor allows editing the scale factor as follows:

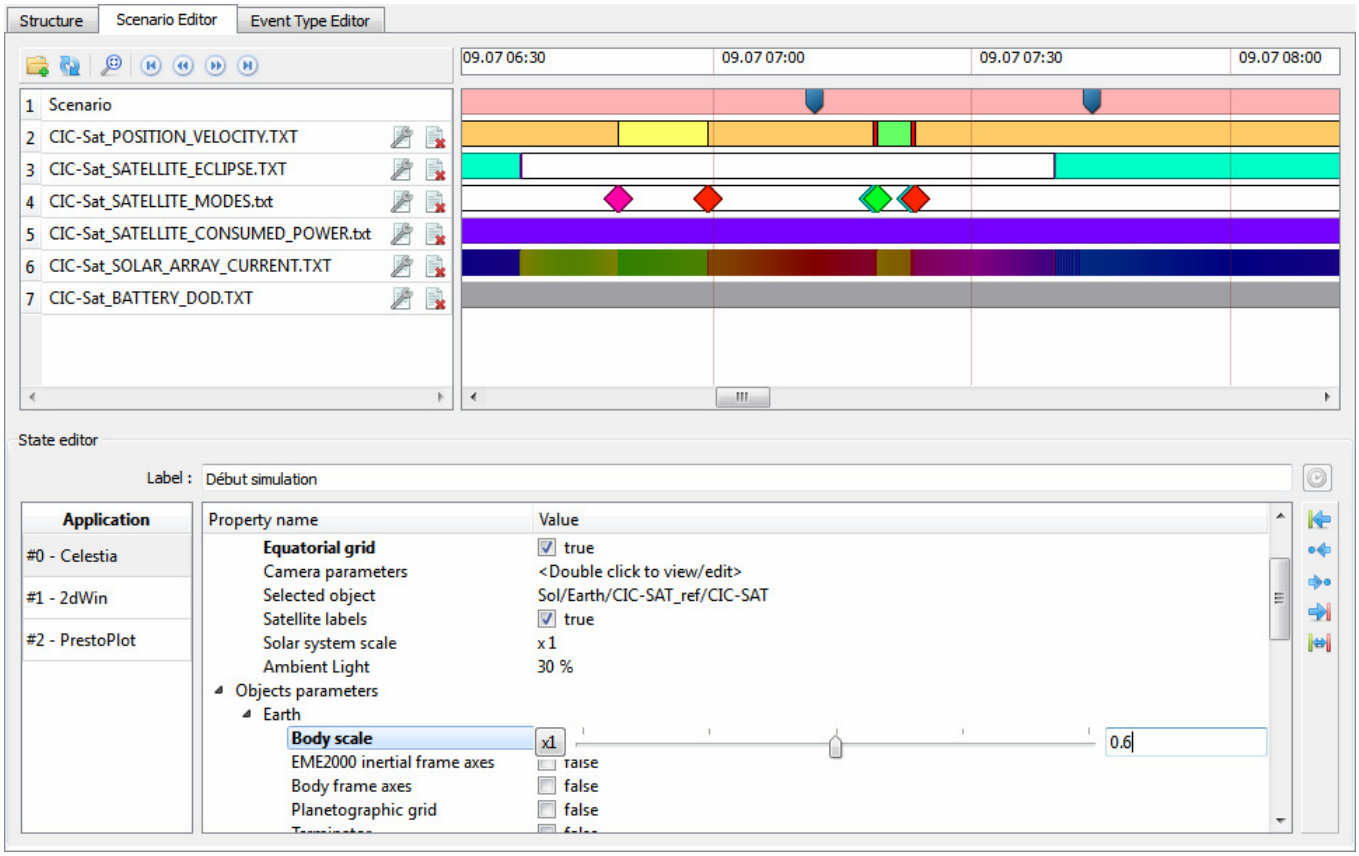
- by directly entering a numerical value in the text field
- by moving the slider left/right to increase/decrease the factor
- by setting the factor to 1 using the **x1** button
- by setting the factor to 100□000 using the **x1e5** button (satellites only)

3.13. SCENARIO IN VTS

The VTS project scenario consists of a set of visualization states applied at precise instants of the visualization.

These states hold visualization properties specific to each client, such as position and orientation of the camera, visibility of satellite components, visibility of the equatorial grid, etc. Special behavior can then be triggered at precise dates through visualization states, e.g. momentarily switching to sensor view when an acquisition occurs on some instrument.

VTS offers two tools to edit a project scenario: the timeline and the view properties editor. In the VTS configuration utility, these tools are available in the Scenario Editor tab.

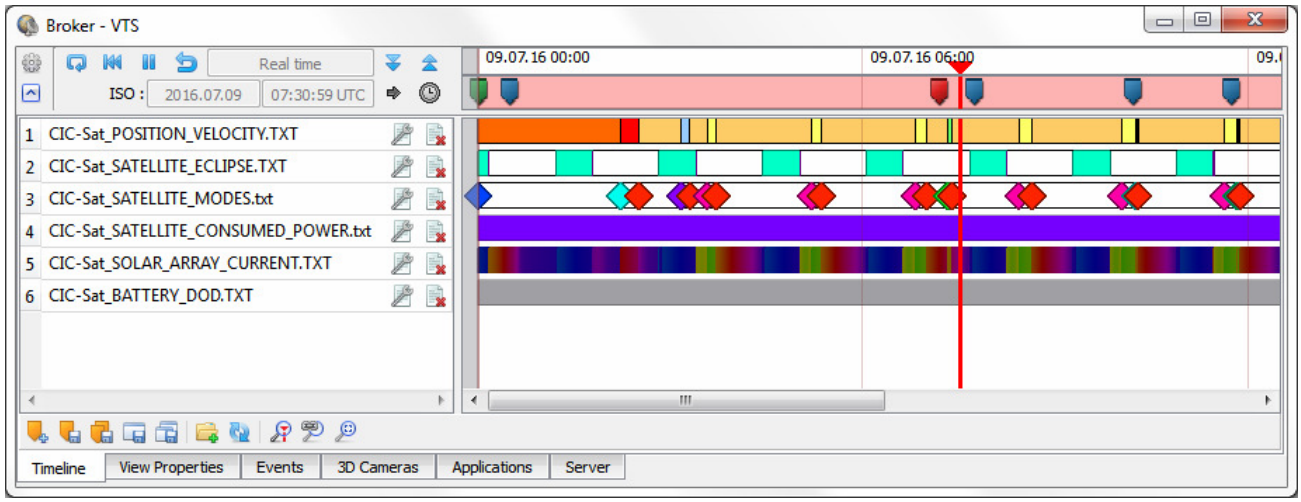


Scenario editor in the VTS configuration utility

In the Broker, they are available in the Timeline and View Properties tab. Parts of the timeline, displaying dates and the project scenario, are always visible in the Broker, no matter which tab is currently selected.

3.13.1. Timeline

The timeline displays and allows selecting, creating and removing visualization states in the project scenario. It also displays the contents of the project's data files and event files.



Timeline in the Broker

The timeline can be zoomed in/out using the mouse wheel, and panned by drag-and-dropping when the mouse cursor has the shape of a hand. This allows for great flexibility during visualization of a project, since the timeline can easily display an overview of the whole project scenario or a detailed view of a small time frame.

Dates outside of the date range of the project are grayed out in the timeline.

In the VTS configuration utility, modifications to the project in the Structure tab are not reflected automatically in the timeline. The Refresh button must be clicked to update the view.

3.13.1.1. Time cursor

The timeline's red time cursor indicates the current visualization date. It can be drag-and-dropped to modify the visualization date. While moving the time cursor, the visualization date is broadcasted synchronously to all client applications.

Double-clicking in the timeline moves the time cursor to the mouse location (and sets the visualization date accordingly). Hovering above the time cursor displays a tooltip indicating the visualization date in ISO format. When zooming the timeline, holding the Ctrl key locks the zoom on the time cursor.

The time cursor is not displayed in the VTS configuration utility, where time does not flow.

3.13.1.2. Project scenario

The project scenario is represented as a red line, labelled Scenario in the VTS configuration utility, and displayed in the compact mode GUI of the Broker.

3.13.1.2.1. Scenario states

Scenario states appear as flags in the project scenario line:

- a blue flag for a regular state not currently selected
- a red flag for the active state (the one displayed in the view properties editor)
- an green flag for the initial state of the visualization
- a grey flag for a disabled state (Broker only)

Also, an orange star appears on states which have been modified but not saved.

In the VTS configuration utility, clicking a state makes it become the active state. In the Broker, the active state is the first enabled state to the left of the time cursor.

When the time cursor moves past a new state in the Broker, it becomes active and its visualization properties are sent to the corresponding client applications. This allows, for example, precisely setting the camera at a user-defined location upon key events of the visualization (instrument acquisition, ground station communication, orbital maneuver, etc.). Note that a disabled state is never active: when disabling the active state, the first enabled previous state becomes active.

The initial state of the visualization is a fictional state mirroring all scenario states prior to the project's start date. It cannot be removed or disabled, and its date is automatically adjusted when the project's start date changes.

3.13.1.2.2. Interacting with the scenario

Right-clicking the scenario displays a context menu with the following actions:

- The **Create state** entry creates a new scenario state at the selected location.
- The **Select current state** entry marks as active the first enabled state to the left of the selected location.

The properties of a new state are copied from the previous state at the date of the new state.

Right-clicking a scenario state displays a context menu with the following actions:

- The **Go to state** entry sets the visualization date to the date of the selected state (Broker only).
- The **Disable state**/**Enable state** entry disables/enables the selected state (Broker only).
- The **Delete state** entry removes the selected state.

Right-clicking on the initial state of the scenario displays a specific context menu:

- The **Go to state** entry sets the visualization date to the date of the selected state (Broker only).
- The **Create state** entry creates a new scenario state at the start of the scenario.

3.13.1.3. Mission events

Mission events attached to project entities are displayed in the first lines of the timeline. Each entity has its own line of events. Refer to the Mission events in VTS chapter for more information on events.

3.13.1.3.1. Graphical representation of events

The header of each line shows the name of the entity for the events on the line.

Each event is represented by its decoration, configured in the event types editor. Refer to the Configuring event types section of the VTS configuration utility user manual chapter for more information.

Each line can be resized by resizing its header's line number. Lines can be reordered by drag-and-dropping their headers' line numbers.

3.13.1.3.2. Interacting with events

Right-clicking on an event displays a context menu with the following actions:

- The **Go to event** entry sets the visualization date to the date of the selected event (Broker only).
- The **Select state for event** entry marks as active the first enabled state to the left of the selected event.
- The **Create state at event** entry creates a new scenario state at the date of the selected event.

3.13.1.4. Project scripts

VTS scripts used by the project are displayed in the timeline. Each script has its own line. Refer to the Scripts and macros in VTS chapter for more information on script files.

3.13.1.4.1. Graphical representation of scripts

The header of each line shows the name of the script.

Script commands are displayed as red diamonds at the date at which they occur.

Each line can be resized by resizing its header's line number. Lines can be reordered by drag-and-dropping their headers' line numbers.

3.13.1.4.2. Interacting with scripts

The following actions are available for script headers:

- Double-clicking the header sets the zoom level and pans the timeline so that the line's width fits in.
- The **File properties...** button opens a pop-up window with information on the script file.
- The **Remove file** button removes the script from both the timeline and the project.

Right-clicking on a script command displays a context menu with the following actions:

- The **Go to command** entry sets the visualization date to the date of the selected command (Broker only).
- The **Select state for command** entry marks as active the first enabled state to the left of the selected command.
- The **Create state at command** entry creates a new scenario state at the date of the selected command.

3.13.1.5. Project data files

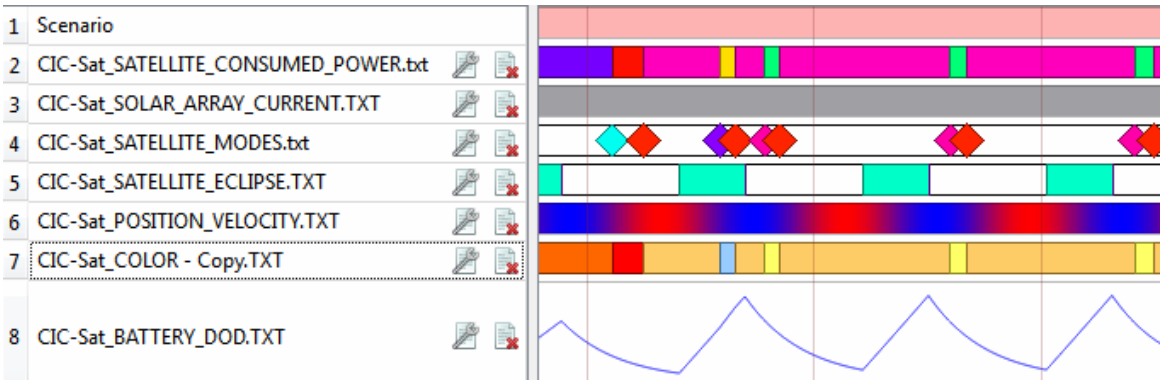
Data files used by the project (position and orientation of satellites, sensors, etc.) are displayed in the timeline. Each file has its own line.

3.13.1.5.1. Graphical representation of files

The header of each line shows the name of the file.

The following display modes are available:

- **Normal** mode: Each value is displayed as a rectangle extending from the date of the value to the date of the next value. Rectangles of identical color indicate identical value. In the screenshot below, this display mode is used for line 2.
- **Block** mode: The whole file is displayed as a single rectangle. This display mode is used for files with more than 300 lines or bigger than 500Kb, so as not to overload the timeline. When the file contents have not been read, the block is colored in gray and no tooltip is available. Otherwise, the block is colored and tooltips are available. In the screenshot below, this display mode is used for line 3.
- **Event** mode: Each value is displayed as a diamond at the date of the value. Diamonds of identical color indicate identical value. In the screenshot below, this display mode is used for line 4.
- **Interval** mode: Identical to **Normal** mode, except that "neutral" values (0 for real or integer data) are not displayed. This mode offers better contrast to visualize the contents of files with on/off status, such as in/out of eclipse status for a satellite, etc. In the screenshot below, this display mode is used for line 5.
- **Gradient** mode: The whole file is displayed as a single rectangle filled with a color gradient indicating the value variations in the file. This mode offers better visualization of the evolution of values in a file, but may not be suited to files of dimension greater than 1. In the screenshot below, this display mode is used for line 6.
- **Color** mode: Identical to **Normal** mode, except that the color of each rectangle is derived by interpreting the corresponding value as a color. This mode allows displaying the contents of color files, and is only available for data which may be interpreted as a color, i.e. files of real or integer data of dimension 3 or 4. In the screenshot below, this display mode is used for line 7.
- **Graph** mode: all values are displayed as a graph. This is a preview of the content of the file and may not replace a proper plotter software such as PrestoPlot. In the screenshot below, this display mode is used for line 8.



Display modes available for data files in the Timeline

In order to avoid long loading times, files bigger than 500Kb are only partially loaded. The full contents will however be read if the user selects a mode other than Block for the file. When hovered, files for which the full contents have been read display a tooltip containing the value at the current location of the mouse.

Each line can be resized by resizing its header's line number. Lines can be reordered by drag-and-dropping their headers' line numbers.

Color files associated with OEM position files are used to color the line of their position file.

User-selected display modes are saved in the VTS project file and restored upon loading of the project in the VTS configuration utility or the Broker. Be aware that this might result in longer loading times if a mode other than Block is selected for files of significant size.

3.13.1.5.2. Interacting with files

The following actions are available for file headers:

- Double-clicking the header sets the zoom level and pans the timeline so that the line's width fits in.
- The **File properties...** button opens a pop-up window with information on the file and parameters controlling its appearance in the timeline.
- The **Remove file** button removes the file from the timeline. Files referenced in the project structure are not removed from the project, they are merely hidden in the timeline and will remain so until they have been added back.

The file properties pop-up, apart from displaying information on the file, offers the following actions:

- The **Edit file...** button opens the file in a text editor.
- The **Display mode** drop-down list allows selecting the display mode for the file.
- The **Merge identical values** option enables/disables the merging of consecutive identical values in the file into a single value item (rectangle or diamond). This is on by default to improve the clarity of the timeline's contents.
- The **Color overlay** option enables/disables the color file overlay for OEM position files.

Note that the number of lines is not displayed in the Properties tab for partially loaded files.

Right-clicking on values in the file's line displays a context menu with the following actions (not available in Block and Gradient modes):

- The **Go to event/Go to interval start/Go to interval end** entry sets the visualization date to the date/start date/end date of the selected diamond/rectangle (Broker only).

- The **Select state for event**/**Select state for interval start**/**Select state for interval end** entry marks as active the first enabled state to the left of the selected diamond/rectangle start/end.
- The **Create state at event**/**Create state at interval start**/**Create state at interval end** entry creates a new scenario state at the date of the selected diamond/rectangle start/end.

3.13.1.6. Timeline toolbar

The timeline's toolbar offers the following actions:

- The **Add files...** button opens a browser to add one or several files (data files or scripts) to the timeline. These files will have to be saved in the project folder if they are located outside of it. Files not part of the structure added this way will be saved and opened on future loads of the project. Files may also be added by drag-and-dropping them onto the timeline.
- The **Refresh timeline** button updates the timeline. The project dates, files (events and data) and the contents of these files are updated. The timeline may only be updated if the project is in a valid state.
- The **Center cursor** button centers the timeline's view on the time cursor (Broker only).
- The **Follow cursor** button automatically locks the view on the time cursor as it moves with the visualization date (Broker only).
- The **Reset view** button zooms in/out the timeline and pans it so that the whole project is displayed.

In the VTS configuration utility, the following buttons allow navigation between scenario states:

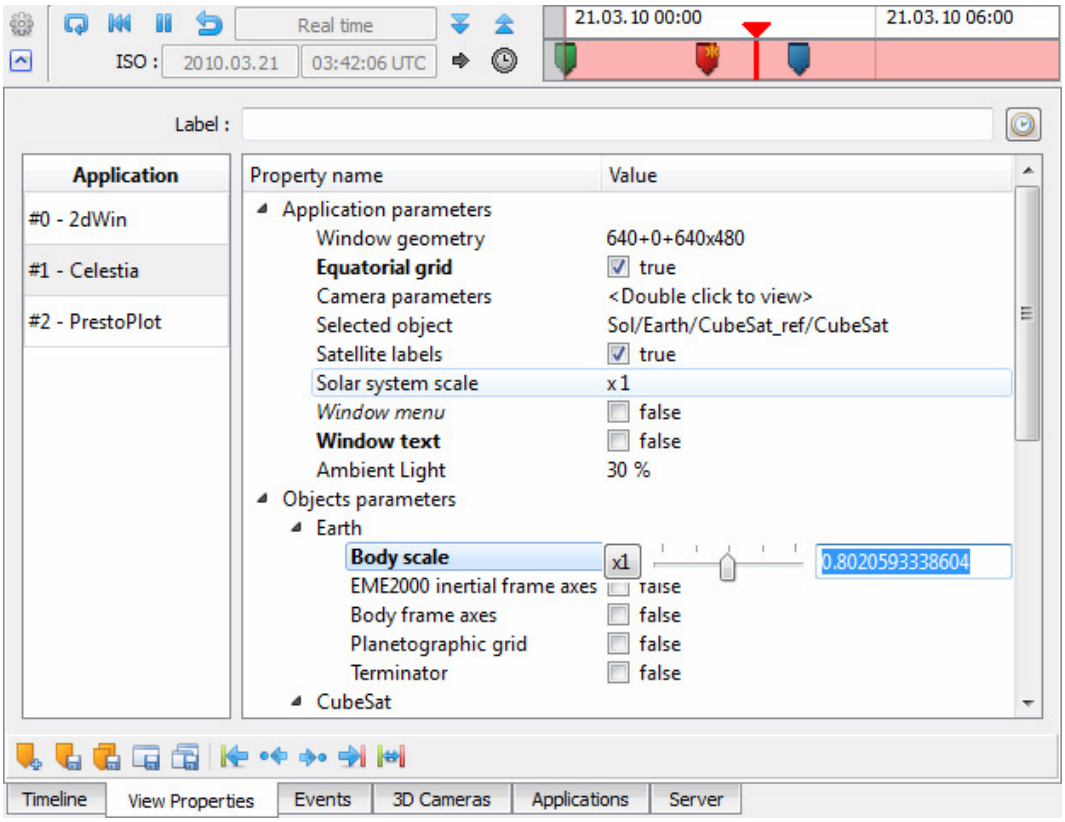
- The **Initial state** button marks the project scenario's initial state as active.
- The **Previous state** button marks the active state's previous state as active.
- The **Next state** button marks the active state's next state as active.
- The **Final state** button marks the project scenario's last state as active.

In the Broker, the following buttons allow saving view properties of client applications into scenario states:

- The **Create state** buttons creates a new scenario state at the current visualization date and fills it with view properties gathered from all client applications. The new state is saved (others are not).
- The **Save current state only** button updates the currently active scenario state with view properties gathered from all client applications. The active state is saved (others are not).
- The **Save all states button** button updates the currently active scenario state with view properties gathered from all client applications, and saves all scenario states.
- The **Save window positions for current state only** button updates the currently active scenario state with the window position of all client applications. The active state is saved (others are not).
- The **Save window positions for whole scenario** button updates all scenario states with the window position of all client applications. All scenario states are saved.

3.13.2. View properties editor

The view properties editor allows editing the view properties of client applications attached to each scenario state.



View properties editor in the Broker

3.13.2.1. Scenario state properties

The Label field displays the label of the current scenario state. It can be modified.

The Date button opens a dialog to view and edit the date of the current scenario date. However, it is not possible to modify the date of the fictional initial scenario state.

The Application list displays a list of all client applications. Selecting a client application in the list displays its view properties in the tree hierarchy to the right. There are two types of view properties: application parameters, which describe global properties specific to the application; and objects parameters, which describe display properties of project entities (central bodies, satellites, sensors, etc.) for the application.

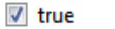
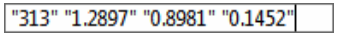
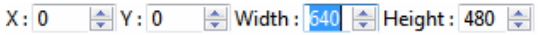
Each property has a label describing its purpose. Clicking or double-clicking a property's value to the right of its label opens an editor to modify the value.

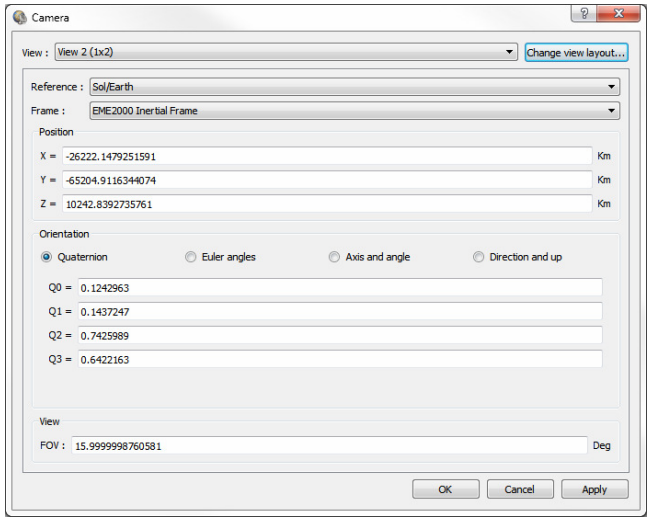
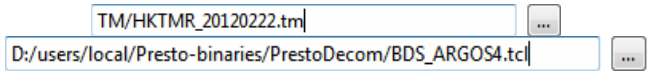
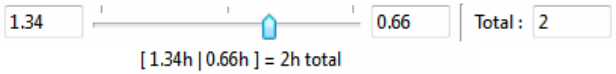
- Properties with bold labels have had their values modified but not saved. Scenario states for which at least one of their properties has had its value modified appear with an orange star in the timeline.
- Properties with italic labels have a unique value for the whole scenario. Modifying their value automatically affects all scenario states.

Each property can be reset to default by right-click. In the Broker, this action can lead to a message to the concerned application.

The table below describes the

property editors associated with the various property types:

Property type	Editor	Description
Boolean		Editor for a boolean property (e.g.: equatorial grid visibility in Celestia). Clicking the checkbox toggles the property.
Integer or real number		Editor for an integer or real number property. The integer or real value can be entered manually or increased/decreased using the buttons.
Character string		Editor for a string property (e.g.: selected object in Celestia). The string must be entered manually.
Character string list		Editor for a string list property (e.g.: visible event types in 2dWin). The string list must be entered manually. Syntax rules of the VTS synchronization protocol apply (except the newline rule). Refer to the Message syntax section of the Synchronization protocol for VTS clients chapter for a list of these rules.
Scale factor		Editor for a scale factor property (e.g.: satellite scale factor in Celestia). The scale factor value can be adjusted by dragging the slider left or right, or entered manually, or reset using the buttons.
Range		Editor for a range property (e.g.: ambient light level in Celestia). The range value can be set by moving the slider, or entered manually, or reset using the button.
Rectangle		Editor for a rectangle property (e.g.: window position and size). The X, Y coordinates and width, height of the rectangle can each be entered manually or increased/decreased using the buttons.

Property type	Editor	Description
Camera		Editor for camera parameters (e.g.: point of view in Celestia). • The camera being configured is selected in the **View** drop-down list. The Celestia view layout can be controlled using the **Change view layout...** button. • The reference frame the camera is attached to is defined by an object selected in the **Reference** drop-down list, and a frame selected in the **Frame** drop-down list. • The position of the camera in its reference frame can be entered in the **Position** group, in kilometers. • The orientation of the camera in its reference frame can be entered in the **Orientation** group, either as a quaternion, as Euler angles (in degrees), as an axis and angle (in degrees), or as a direction and up vector. • The field of view of the camera can be entered in the **FOV** field, in degrees.
File		Editor for file properties. The file path (either relative or absolute) can be entered in the text field or selected through a file browser using the button.
Time Window		Editor for a time window. It represents the time distribution before and after a dated event. Durations are in hours.

Further documentation on the properties of standard client applications can be found in the Synchronization protocol for VTS clients chapter. In particular, the Messages received by Celestia and Messages received by 2DWin sections describe properties of the Celestia and 2DWin client applications.

3.13.2.2. View properties editor toolbar

The view properties editor offers buttons to propagate view properties between scenario states. Upon selection of one or several properties in the tree hierarchy, these buttons allow copying the selected properties for the same application in one or several other scenario states:

- The **Propagate selection to all previous states** button copies the selected view properties to all previous scenario states.
- The **Propagate selection to previous state** button copies the selected view properties to the previous scenario state.
- The **Propagate selection to next state** button copies the selected view properties to the next scenario state.
- The **Propagate selection to all next states** button copies the selected view properties to all next scenario states.
- The **Propagate selection to all states** button copies the selected view properties to all scenario states.

These actions allow easily setting a property value for multiple scenario states. Several properties can be selected by holding down the Ctrl or Shift keys while selecting the properties to propagate. If a tree node is selected, all properties beneath it will be propagated.

In the Broker, the view properties editor also offers the same buttons as those available to save states in the timeline.

3.14. MISSION EVENTS IN VTS

Mission events in VTS are timestampped punctual events attached to visualization entities. Currently, events may only be attached to satellites.

Events are displayed in the project timeline and in compatible client applications. Currently, the only standard VTS client application with events capability is 2DWin.

An event is defined by its event type and timestamp. Text metadata may also be attached to an event.

3.14.1. Event type

The event type is a category describing the event that occurs at the dates of instances of this type. For example, events of the "OEF/EARTH_IN_SENSOR_END (2)" event type are orbit events at the dates when the Earth leaves the field of view of the second star tracker.

Event types in VTS are organized hierarchically. An event type full name includes the names of all its parent types, separated with slashes, e.g. "PASS/KRU/TC_EMISSION_END".

3.14.2. CIC/CCSDS event files

Events are described in CIC/CCSDS files in MEM format. For more information on the CIC/CCSDS file format, refer to the CIC/CCSDS data files in VTS chapter.

A CIC/CCSDS event file must have the following characteristics:

- The *USER_DEFINED_PROTOCOL* must be **NONE**.

- The *USER_DEFINED_CONTENT* is the top-level event type name for all events in the file. It will be implicitly prepended to the event type names found in the file.
- The *USER_DEFINED_SIZE* must be **1** or more. The first column (after the date) must be the event type full name (not including the top-level type, *USER_DEFINED_CONTENT*). All other columns will be used as text description/metadata for the event.
- The *USER_DEFINED_TYPE* must be **STRING**.
- The *USER_DEFINED_UNIT* must be **[n/a]**.

The following is a sample CIC/CCSDS event file:

```
CIC_MEM_VERS   = 1.0
CREATION_DATE  = 2013-07-18T16:19:51
ORIGINATOR     = PROTON

META_START

COMMENT PROTON OEF file
COMMENT Misc columns: PSO Lat Long Cycle Orbit

OBJECT_NAME = PLEIADES
OBJECT_ID   = PHR

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = OEF
USER_DEFINED_SIZE = 6
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

TIME_SYSTEM = UTC

META_STOP

56273 0.000000        "DAS_Update_TC_CHCOMRLDDAS"   "25.7" "25.39" "-26.5"    "8"  "272"
56273 126.809000      "AUS/0_DEG_AOS"                     "33.44" "33.05" "-28.48"  "8"  "272"
56273 282.221000      "AUS/MAX_ELEVATION_PASS)"     "42.9" "42.38" "-31.3"    "8"  "272"
56273 438.342000      "AUS/0_DEG_LOS"                     "52.4" "51.68" "-34.89"  "8"  "272"
56273 453.909000      "STH/0_DEG_AOS"                     "53.35" "52.6"         "-35.32"  "8"  "272"
56273 510.640000      "STH/PHYSICAL_AOS"            "56.8" "55.95" "-36.99"  "8"  "272"
56273 566.297000      "STH/MAX_ELEVATION_PASS"      "60.18" "59.21" "-38.91"  "8"  "272"
56273 616.599000      "KRN/0_DEG_AOS"                     "63.24" "62.14" "-40.93"  "8"  "272"
56273 663.121000      "KRN/PHYSICAL_AOS"            "66.07" "64.82" "-43.15"  "8"  "272"
56273 678.672000      "STH/PHYSICAL_LOS"            "67.01" "65.71" "-43.98"  "8"  "272"
56273 678.672000001   "STH/0_DEG_LOS"                     "67.01" "65.71" "-43.98"  "8"  "272"
56273 769.656000      "AUTONOMOUS_MODE_GAP/SUP_START"      "72.54" "70.8"         "-50.17"  "8"  "272"
56273 780.478000      "KRN/MAX_ELEVATION_PASS"      "73.2" "71.39" "-51.11"  "8"  "272"
56273 803.998000      "EARTH_IN_SENSOR_START (3)"   "74.63" "72.66" "-53.34"  "8"  "272"
56273 808.998000      "SUN_IN_SENSOR_START (3)"     "74.93" "72.92" "-53.85"  "8"  "272"
56273 872.998000      "SUN_IN_SENSOR_END (3)"             "78.82" "76.19" "-62.02"  "8"  "272"
56273 875.835000      "KRN/PHYSICAL_LOS"            "78.99" "76.33" "-62.47"  "8"  "272"
56273 908.478000      "NPL/0_DEG_AOS"                     "80.98" "77.86" "-68.28"  "8"  "272"
56273 935.329000      "NPL/PHYSICAL_AOS"            "82.61" "79."  "-74.17"  "8"  "272"
56273 942.531000      "AUTONOMOUS_MODE_GAP/SUP_END" "83.05" "79.29" "-75.95"  "8"  "272"
56273 944.742000      "KRN/0_DEG_LOS"                     "83.18" "79.37" "-76.51"  "8"  "272"
56273 947.195000      "SHADOW_PENOMBRA"             "83.33" "79.47" "-77.15"  "8"  "272"
56273 957.008000      "PENOMBRA_LIGHT"              "83.93" "79.83" "-79.82"  "8"  "272"
```

3.14.3. Event decorations

Event decorations define the appearance of events in the project timeline and in compatible client applications.

Each event type may have its own custom decoration, in order to be easily distinguishable from other event types. Note that all events of a single event type have the same appearance.

Thanks to the hierarchy of event types, event decorations are inherited: all children type of an event type share the same decoration as their parent type. This inheritance mechanism can be overridden by specifically customizing the decoration of an event type.

For more information on the specifics of event decoration configuration, refer to the Configuring event types section

of the VTS configuration utility user manual chapter.

3.15. POIS AND ROIS IN VTS

POIs and ROIs in VTS are files defining Points and Regions of Interest for the visualization. They are displayed in 2D and 3D applications that support them. Both 2dWin and Celestia support displaying POIs and ROIs.

3.15.1.1. CIC/CCSDS POI and ROI file format

POIs and ROIs are defined in CIC/CCSDS files in MPM format. For more information on the CIC/CCSDS file format, refer to the CIC/CCSDS data files in VTS chapter.

A CIC/CCSDS POI or ROI file must have the following characteristics:

- The *USER_DEFINED_PROTOCOL* must be **NONE**.
- The *USER_DEFINED_CONTENT* may be arbitrary.
- The *USER_DEFINED_SIZE* must be **2**, or **3** for POIs only.
- The *USER_DEFINED_TYPE* must be **REAL**, or **STRING** for POIs only with a size of 3.
- The *USER_DEFINED_UNIT* must be **[deg]**, or **n/a** for POIs only with a size of 3.

The data values in the file are coordinates in latitude/longitude.

In ROI files, the special data value 180 180 is interpreted as a separator and will create a new polygon with the following sequence of coordinates.

A single POI file may define multiple points of interest. In POI files with 3 columns, the last column defines a label for the corresponding coordinates.

3.15.1.2. Sample POI file

The following is a sample CIC/CCSDS POI file:

```
CIC_MPM_VERS = 1.0
CREATION_DATE = 2014-04-23T15:25:04.268055
ORIGINATOR = VTS

META_START

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = POINTS OF INTEREST
USER_DEFINED_SIZE = 3
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

META_STOP

48.861348   2.345248   "Siège social"
48.843445   2.390097   DLA
43.561948   1.481500   CST
 5.208395 -52.775477   CSG
```

3.15.1.3. Sample ROI file

The following is a sample CIC/CCSDS ROI file containing two polygons:

```
CIC_MPM_VERS = 1.0
CREATION_DATE = 2014-04-23T15:25:04.268055
ORIGINATOR = VTS
```

```
META_START

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = REGION OF INTEREST
USER_DEFINED_SIZE = 2
USER_DEFINED_TYPE = REAL
USER_DEFINED_UNIT = [deg]

META_STOP

-30 30
30 30
0 0
180 180
0 0
-25 -25
25 -25
```

3.16. SCRIPTS AND MACROS IN VTS

Scripts and macros in VTS are commands from the VTS synchronization protocol, which are sent by the Broker to client applications during visualization.

Scripts are timestamped commands sent at predefined dates, while macros are non-timestamped commands sent in a batch upon macro execution. When the visualization date reaches the timestamp of a script command, the command is sent to all client applications specified in the recipient field of the script line.

Scripts are displayed in the project timeline. Macros are listed in the Broker menu. For more information on how to use scripts and macros in VTS, refer to the Timeline section in the Scenario in VTS chapter, and the Broker menu section in the Broker user manual chapter.

3.16.1. CIC/CCSDS script and macro file format

Scripts are written as CIC/CCSDS files in MEM format. Macros are written as CIC/CCSDS files in MPM format. For more information on the CIC/CCSDS file format, refer to the CIC/CCSDS data files in VTS chapter.

A CIC/CCSDS script or macro file must have the following characteristics:

- The *USER_DEFINED_PROTOCOL* must be **NONE**.
- The *USER_DEFINED_CONTENT* must be **SCRIPT** for a script, **MACRO** for a macro.
- The *USER_DEFINED_SIZE* must be **2**. The first column (after the date in the case of scripts) must be the recipient specification for the command (see below). The second column must be the command to send (see below).
- The *USER_DEFINED_TYPE* must be **STRING**.
- The *USER_DEFINED_UNIT* must be **[n/a]**.

3.16.2. Command recipient specification

The recipient for a script or macro command may be specified using any of the following:

- ***** indicates that the command shall be sent to all clients
- **broker** indicates that the command shall be sent to the Broker itself
- ***<Name>*** indicates that the command shall be sent to all client applications with the specified name

- ***<ID>*** indicates that the command shall be sent to the client with the specified ID
- ***<Name>:<#>*** indicates that the command shall be sent to the *n*-th instance of the client application with the specified name
- ***:<#>*** indicates that the command shall be sent to the *n*-th client application (using the order of the **Applications** tab of the Broker)

3.16.3. Command contents

Refer to the Synchronization protocol for VTS clients chapter for full details on the syntax of commands, and for a list of available commands in the standard client applications.

In order to allow "customized" commands to be sent to several client applications with a single line in a script or macro file, some special strings in script and macro file commands are automatically replaced before sending the command to clients:

- **%APPNAME%** gets replaced with the recipient client application name and ID, in the following format: *<Name>-<ID>*
- **%DATE%** gets replaced with the sending date, in the following format: *yyyy-MM-dd--hh-mm-ss-zzz*
- **%COUNT%** gets replaced with a sequence number, counting from 0 and shared between all script files, macro files and client applications in the project
- **%PROJECT%** gets replaced with the full path to the project folder

Note that when taking screenshots in Celestia, the %PROJECT% string is required at the beginning of the screenshot name to store the screenshot in the project folder. Otherwise, Celestia stores the screenshot in the Apps/Celestia/bin directory of the main VTS folder.

3.16.4. Sample script file

The following is a sample CIC/CCSDS MEM script file:

```
CIC_MEM_VERS   = 1.0
CREATION_DATE  = 2014-02-18T16:19:51
ORIGINATOR     = VTS

META_START

OBJECT_NAME = SCRIPT_SCREENSHOT
OBJECT_ID   = SCR001

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = SCRIPT
USER_DEFINED_SIZE = 2
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

TIME_SYSTEM = UTC

META_STOP

55276  3000 celestia   "CMD PROP equatorialgrid true"
55276  3001 *          "CMD SERVICE TakeScreenshot %PROJECT%/%DATE%_%APPNAME%_%COUNT%"
55276  3002 celestia   "CMD PROP equatorialgrid false"
55276  3003 *          "CMD SERVICE TakeScreenshot %PROJECT%/%DATE%_%APPNAME%_%COUNT%"
55276  3004 2dwin:0    "CMD STRUCT OrbitVisible \"Sol/Earth/CubeSat\" false"
55276  3005 *          "CMD SERVICE TakeScreenshot %PROJECT%/%DATE%_%APPNAME%_%COUNT%"
```

3.16.5. Sample macro file

The following is a sample CIC/CCSDS MPM macro file:

```
CIC_MPM_VERS   = 1.0
CREATION_DATE  = 2014-06-12T16:54:12
ORIGINATOR     = VTS

META_START

OBJECT_NAME = MACRO_4SPLIT
OBJECT_ID   = MAC001

USER_DEFINED_PROTOCOL = NONE
USER_DEFINED_CONTENT = MACRO
USER_DEFINED_SIZE = 2
USER_DEFINED_TYPE = STRING
USER_DEFINED_UNIT = [n/a]

META_STOP

:1 "CMD PROP WindowGeometry 0 0 640 480"
:2 "CMD PROP WindowGeometry 640 0 640 480"
:3 "CMD PROP WindowGeometry 0 480 640 480"
:4 "CMD PROP WindowGeometry 640 480 640 480"
```

3.17. VTS BROKER

The Broker is the core application during visualization of a VTS project. It controls and synchronizes all client applications in time.

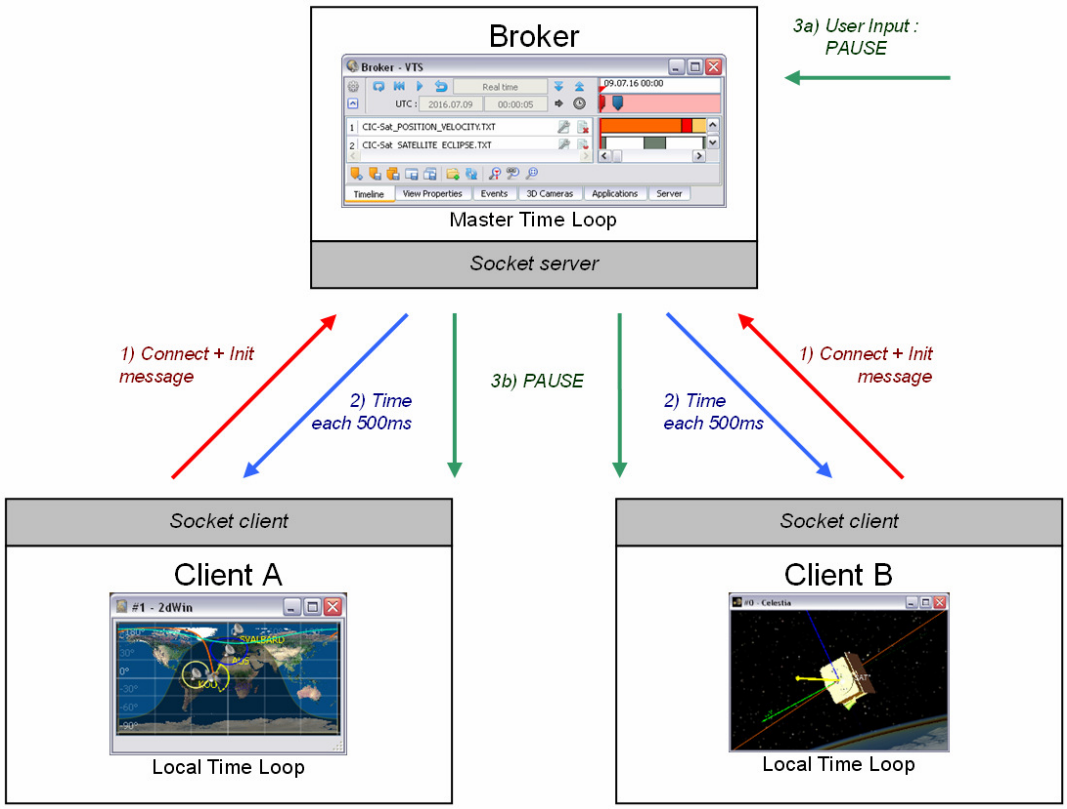
Client applications see the Broker as a "socket" server to which they connect and with which they communicate. Each client has a dedicated communication channel with the Broker. When an action or event within the Broker requires communication with its client applications, the Broker decides whether to broadcast the corresponding message to all clients or only to specific ones. Client-Broker communications follow a protocol based on text socket, described in more detail in the Synchronization protocol for VTS clients section.

Messages from the Broker to its clients concern:

- time synchronization
- view controls
- camera controls
- events management
- scripted commands
- context saving
- relayed messages from other clients

Some messages expect reply messages from the clients. Refer to the Synchronization protocol for VTS clients chapter for more information.

The Broker also has the responsibility of managing the life cycle of its clients (either configured in the VTS project or started by user interaction): spawning new clients, relaying client log messages to the user, displaying connection status about the clients, informing the user of client terminations (or crashes). External clients not started by the Broker obviously fall outside of this responsibility.



Basic synchronization between Broker and clients

4. FILES USED BY VTS

4.1. DATA DESCRIPTION FOR VTS PROJECTS

4.1.1. Hierarchy of a project folder

A VTS project consists in a folder containing one or several project files (.vts file extension), and a set of data files used by client applications during visualization. The project folder is the root folder containing the project file.

The project file contains the description in standard XML format of all the objects in the project. This description refers to data files as paths relative to the project folder. Portability concerns of project folders dictate that data files must reside within sub-folders of the project folder. Should the user select data files outside of the project folder, VTS will automatically offer to copy these files inside the project folder.

Remarks:

- It is recommended to create one folder per VTS project, so that modifying a data file for a given project will not silently affect other projects.
- The project folder can be stored on any device (hard disk drive, USB stick, CD) and must not necessarily be located in a sub-folder of the main VTS folder.
- Some client applications require the availability of data located in folders following a strict nomenclature. The README files for these applications describe their requirements.

4.1.2. 2D icons and textures

Objects projected on the main surface in 2D views are displayed as icons. These icons may be in any of the following formats: .bmp, .gif, .ico, .jpg, .mng, .pbm, .pgm, .png, .ppm, .svg, .svgz, .tga, .tif, .tiff, .xbm, .xpm. Icon files are also used for event decorations. The same formats are supported.

Custom textures for central bodies may also be in any of the above formats.

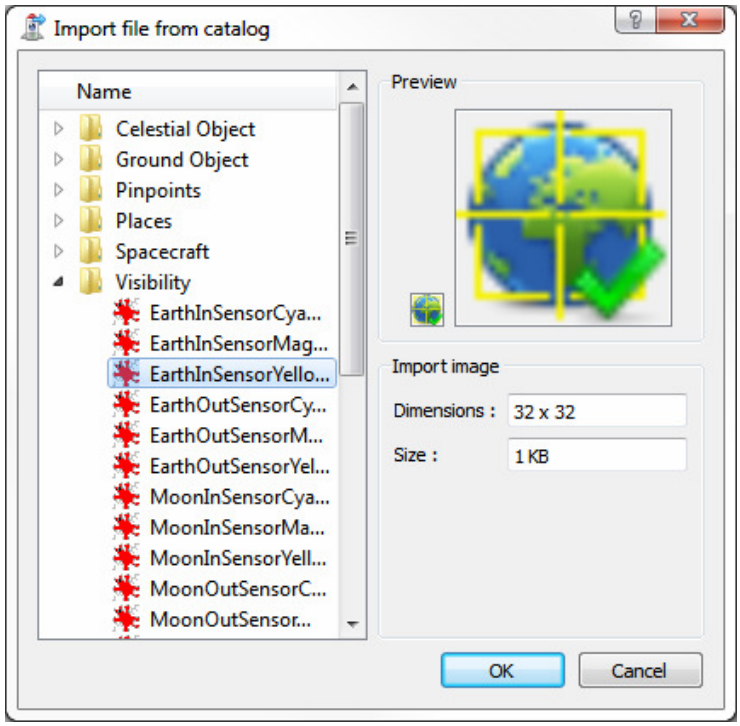
4.1.3. 3D models and textures

The description of a satellite refers to 3D model files. Refer to the 3D file format in VTS chapter for more information.

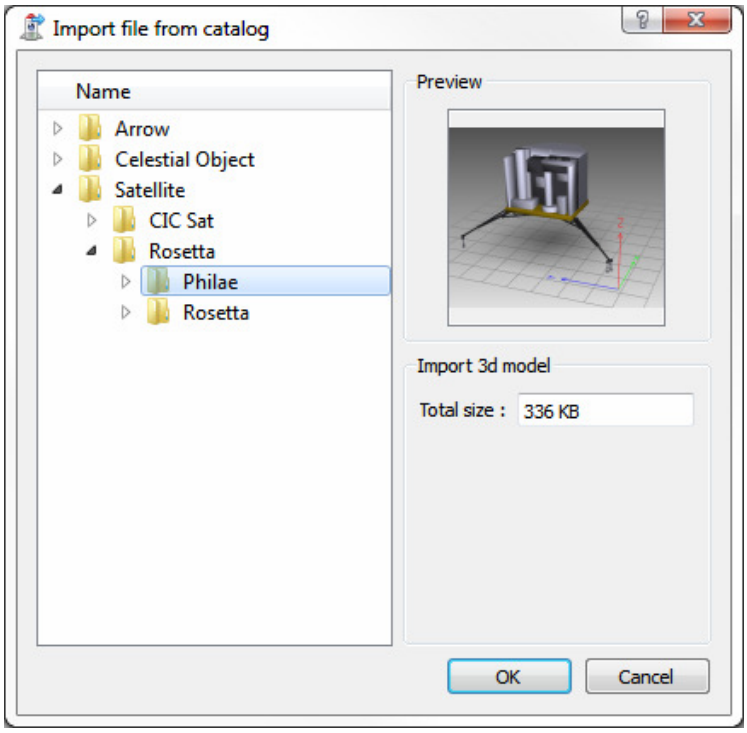
4.1.4. Importing 2D icons and 3D models from the catalog

VTS supplies a catalog of icons and 3D model. Clicking on the import file from catalog  button next to an icon or model browse... button will open a dialog window. Validating the import will offer to copy these files inside the project folder.

Concerning 3D models, a folder contains only one 3DS file and its textures files. Importing the model will import all the files of the folder at once.



Importing an icon from the catalog



Importing a model from the catalog

4.1.5. The CIC/CCSDS file format

This file format is used for various text data, e.g. position and attitude ephemerides. Refer to the CIC/CCSDS data files in VTS chapter for more information.

4.2. 3D FILE FORMAT IN VTS

VTS relies on the 3ds file format, from the Autodesk 3DS Max software suite. This is a widely-used file format for 3D files.

4.2.1. File format of textures

3ds files can make use of external texture files. These must be in bmp format, and be located in the same folder as the 3ds file.

4.2.2. Origin of the reference frame for the 3D file

The position and orientation of objects in 3D views depends on their reference frames. Refer to the following pages:

- Position of objects in VTS
- Orientation of objects in VTS

When position and orientation values are null, the object's local frame is aligned with its reference frame. VTS offers two options to define the local frame of a 3D object:

- Either use the 3D coordinates contained in the *3ds* file. The origin of the local frame corresponds to the point of null coordinates in the file.
- Either define the origin of the local frame as the center of the *3ds* file object's bounding box.

These options can be selected in the VTS configuration utility, independently for each 3D object. The "Use 3ds coordinates" setting must be checked to enable use of the 3D file's origin, unchecked to use the geometric center of the object as origin. Refer to the VTS configuration utility user manual chapter for more information.

It must be well understood that these options make it easier to define satellites composed of multiple components. When all 3D files for a satellite's components and sub-components are extracted from a single master 3ds file, all those files use the same local frame and thus the coordinates can be read directly from the files. The SMOS example supplied with VTS perfectly illustrates this situation.

4.2.3. Dimensions and units

As above, the size of a 3D object can either be read as is from the 3ds file, or defined by VTS. This choice also depends on the option used for the origin of the local frame.

When the origin is the geometric center of the object (the "Use 3ds coordinates" option is unchecked), the size of the object is defined in the VTS configuration utility. The specified size, in meters, is used as the dimension of the smallest bounding box for the object.

When the origin is taken from the 3ds file, the specified size is used as an hint of the object's size, for the 3D views to optimize its display. The given size should be accurate within a factor of 2 (hence the Approximate size label). The user must also specify the unit of the 3ds file's coordinates. The default unit is the meter.

4.3. CIC/CCSDS DATA FILES IN VTS

4.3.1. References

The data files used by VTS and its client applications are written in the CIC/CCSDS format, a column-based text format derived from the CCSDS OEM, AEM and MPM formats.

This format is described in the reference document CIC exchange protocol v1.0 (DCT/DA/PA - 2009.0021267). Good knowledge of this document is mandatory when generating or altering VTS data files.

4.3.2. Summary

Here are some main features of the CIC/CCSDS file format:

- Standardized header depending on file type
- Data as timestamped lines
- MJD date format with 2 fields respectively for days and seconds (see the Date formats in VTS section)
- Satellite position in kilometers
- Orientation as quaternions from EME2000 towards satellite local frame

5. STARTING VTS

5.1.1. Starting the VTS configuration utility

The VTS toolkit can by started by double-clicking the startVTS.exe file under Windows, or by executing the `./startVTS` command under Linux. This displays the main window of the VTS configuration utility. It allows creating a project by setting up the entities to be visualized: satellites, sensors, ground stations, and client applications.

If the launcher is executed from the command line with the `--project <ProjectFile.vts>` argument, the VTS configuration utility automatically loads the given project on startup.

5.1.2. Starting the visualization from the command line

The visualization can be started automatically from the command line via the launcher. The Broker then opens and starts the visualization, without going through the VTS configuration utility.

In batch mode, the following arguments are mandatory:

- *--batch* instructs the launcher to start the Broker directly
- *--project <ProjectFile.vts>* specifies the project file to load. The path can be either relative or absolute.

Other command-line arguments for the Broker can also be given in batch mode.

Sample command line:

```
startVTS.exe --batch --project C:\Project\CubeSat.vts
```

On Linux only, the version of VTS can be obtained with the command :

```
./startVTS --version
```

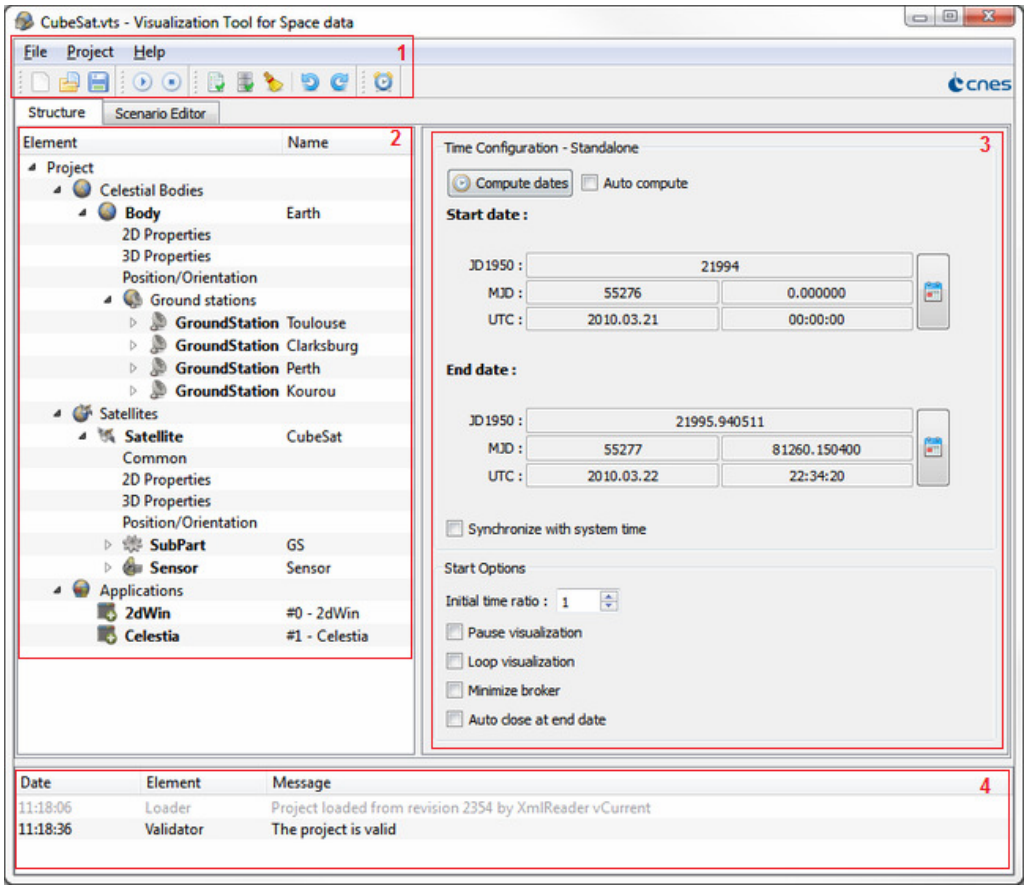
6. VTS APPLICATIONS USER MANUALS

6.1. VTS CONFIGURATION UTILITY USER MANUAL

6.1.1. User interface description

The VTS configuration utility is the entry point for the VTS toolkit. It allows creating and editing VTS projects, as well as starting the visualization of a project.

A project is composed of a set of configurable entities such as satellites, central bodies, ground stations, and client applications. These entities will be used during the visualization phase. It is also possible to configure a scenario of view properties which will be played back during the visualization. The following paragraphs describe in detail how to edit a VTS project.



User interface of the VTS configuration utility

The user interface is composed of:

- A menu and toolbar (1)
- A pane containing a tree hierarchy of the project's entities (2)
- A pane displaying properties of the currently selected entity in the project hierarchy (3)

- A text area for notifications (warnings, errors, etc.) (4)

The Scenario Editor tab features all the tools required to edit the project's scenario. These tools are detailed in the chapter about concepts specific to VTS, in the Scenario in VTS section. The Event Type Editor tab allows customizing the appearance of mission events attached to the project's satellites.

6.1.2. Toolbar actions

The actions below are available in the toolbar of the VTS configuration utility:

- **New Project** button (*Ctrl+N*): create a new project
- **Open Project** button (*Ctrl+O*): load an existing project
- **Save Project** button (*Ctrl+S*): save the current project
- **Run** button (*Ctrl+R*): start the visualization
- **Stop** button: stop the current visualization
- **Check project** button: check that the project is valid
- **Check project and data files** button: check that the project and its data files are valid
- **Clear Logger** button: clear all messages in the notification pane (4)
- **Undo** button (*Ctrl+Z*): undo the previous modification of the project
- **Redo** button (*Ctrl+Y*): redo the next modification of the project (only available with a previously undone modification)
- **Start Propagator** : start the Propagator

A popup window displaying the history of all modifications to the current project can be displayed with the Ctrl+Alt+Z keyboard shortcut.

6.1.2.1. Validity of the project

The Check project action checks the following items:

- Applications check
  - Client applications of the project exist
  - Launchers exist for each of those client applications
- Central bodies check
  - At least one central body is defined
  - All central bodies have unique names
  - Central body textures are valid
  - If applicable, all 3D models used by central bodies exist
  - If applicable, all data files used by central bodies are valid (shallow check)
  - All ground stations on a single body have unique names

    - All POI on a single body have unique names
    - All data files used by POI are valid (shallow check)
    - All ROI on a single body have unique names
    - All data files used by ROI are valid (shallow check)
- Satellites check
    - All satellites have unique names
    - All satellites are attached to a central body
    - All 3D models used by satellites exist
    - If applicable, all data files used by satellites are valid (shallow check)
    - Components check
        - All components of a single satellite have unique names
        - All 3D models used by components exist
        - If applicable, all data files used by components are valid (shallow check)
        - Sub-components check
    - Sensors check
        - All sensors on a given satellite have unique names
        - If applicable, all data files used by sensors are valid (shallow check)
- Dates check
    - Project start date is before its end date
    - All data files used by the project cover the time range of the project
    - Catalog ephemerides for default bodies used in the project cover the time range of the project
- Additional checks
    - All additional files displayed in the timeline are valid (shallow check)
    - All script files used by the project are valid (shallow check)

Checks on data files are shallow, i.e. only the header and file metadata are checked:

- No errors in the header
- Data type described in the metadata matches what VTS expects
    - Data type
    - Data dimension
    - Data unit

6.1.2.2. Validity of the project and its data files

The Check project and data files action covers the perimeter of the Check project action, and also ensures the contents of all data files can be loaded without errors.

6.1.3. Managing a project

6.1.3.1. Creating a project

By definition, the project folder is the folder containing the current project file (.vts). This folder also contains data referenced by the project. A VTS project file is an XML file describing the entities of the visualization, the start and end dates of the visualization, the client applications for the visualization, and the states of the visualization scenario. While editing a project, if the user selects some data or 3D file outside the project folder, a popup will offer to copy the file under the project folder.

Several project files may reside in a common folder. However caution must be taken when altering a shared data file, as this will impact all projects using it.

A new project can be created by doing the following:

- In the toolbar, click the **New project** button, or in the **File** menu click the **New project** entry.
- The **Create a new project** dialog opens, asking for a location to save the project. Select a folder which will contain all files related to the project, fill in the project's name, and click **Save**.

6.1.3.2. Opening a project

An existing project can be opened by doing one of the following:

- Select the project file on disk
  - In the toolbar click the **Open project** button, or in the **File** menu click the **Open project** entry.
  - The **Open project file** dialog opens. Select a project file (*.vts*) and click **Open**.
- Select the project file in the list of the last 15 project files available in the **File** menu.
- Drag-and-drop the project file from a file browser onto the window of the VTS configuration utility.

Opening a project file by double-clicking on it is currently not supported, due to the VTS toolkit being portable.

6.1.3.3. Saving a project

The current project can be saved to disk by doing one of the following:

- In the toolbar click the **Save** button, or in the **File** menu click the **Save** entry.
- To save the project under a different name, click the **Save as...** entry in the **File** menu and modify the name of the project file. Beware that if the project is saved under a different folder, all relative paths in the project will become erroneous.

6.1.3.4. Exporting an archived project

The current project can be exported as a zip archive by clicking on the Create a Project Archive... entry in the File menu. The .vts project file, all the data files in use, all the symbol files in use, and all the 3ds model files in use will

be archived. Notice that all the image files (bmp, png or jpg) found in the used models folders will be exported as VTS doesn't know which texture files are effectively reference by the models files.

6.1.3.5. Visualizing a project

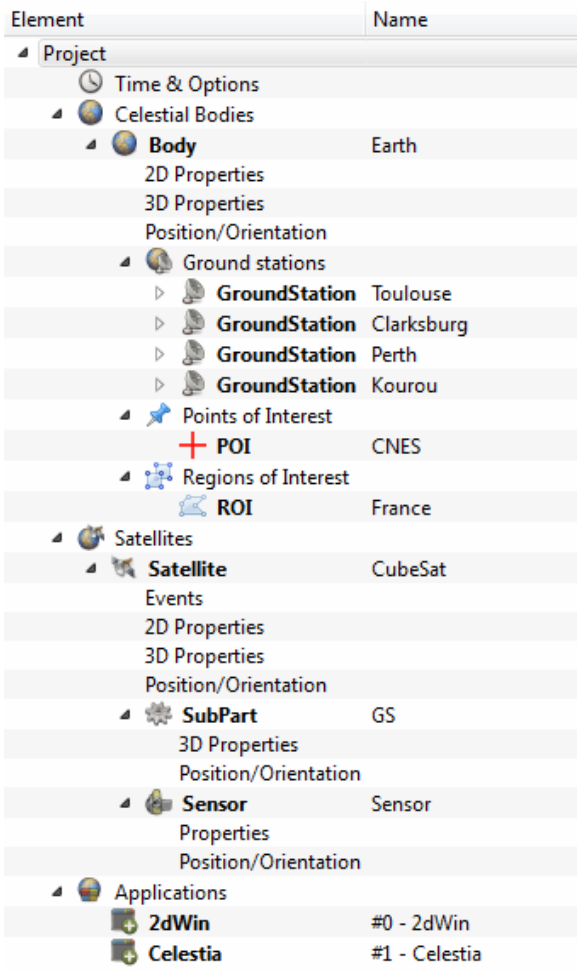
The current project can be visualized by doing the following:

- Start the visualization by clicking the **Run** button in the toolbar, or the **Run** entry in the **Project** menu.

When the visualization ends, a popup offers to load any changes made to the project through the Broker during visualization. These changes can then be saved or discarded.

6.1.3.6. Creating entities

The project hierarchy describes the project's entities in a hierarchical fashion. The hierarchy can be altered through context-sensitive menus.



Projet hierarchy

The following actions can be accessed by right-clicking the project hierarchy:

- on any item
  - *Collapse* folds the selected tree item

  - *Expand* unfolds the selected tree item
  - *Copy* puts a copy of the selected tree item in the clipboard
  - *Paste* pastes a previously copied item onto the selected tree item
- on the **Project** item
  - *View XML* opens the project file in the text editor associated with the *.vts* file extension
  - *Open project location* opens a file browser in the project folder
- on the **Celestial Bodies** item
  - *Add Body* adds a new body to the list of central bodies
- on a **Body** item
  - *Remove* removes the selected body
- on a **Ground stations** item
  - *Add GroundStation* adds a new ground station to the current body
- on a **Ground station** item
  - *Remove* removes the selected ground station
- on a **Points Of Interest** item
  - *Add Point Of Interest* adds a new set of points of interest to the current body
- on a **POI** item
  - *Remove* removes the selected set of points of interest
- on a **Regions Of Interest** item
  - *Add Region Of Interest* adds a new region of interest to the current body
- on a **ROI** item
  - *Remove* removes the selected region of interest
- on the **Satellites** item
  - *Add Satellite* adds a new satellite to the list of satellites
- on a **Satellite** item
  - *Add SubPart* adds a new component to the selected satellite
  - *Add Sensor* adds a new sensor to the selected satellite
  - *Remove* removes the selected satellite
- on a **SubPart** item
  - *Add SubPart* adds a new sub-component to the selected component
  - *Add Sensor* adds a new sensor to the selected component

- *Remove* removes the selected component

- on a **Sensor** item
  - *Remove* removes the selected sensor
- on the **Applications** item
  - *Add Application* adds a new client application to the project (selected in the dropdown menu)
- on any client application item
  - *Remove* removes the selected client application

The Del keyboard shortcut can be used to remove an item from the hierarchy, if the selected item can be removed.

6.1.3.7. Copy-pasting entities

The copy-paste feature can be used to quickly create entities in the project hierarchy.

An entity can be copied through the clipboard either through the Ctrl+C keyboard shortcut, or through the Copy entry in the context menu. Central bodies, satellites, components, sensors, ground stations, points and regions of interest can be copied.

A previously copied entity can be pasted through the Ctrl+V keyboard shortcut, or through the Paste entry in the context menu, once a compatible destination entity has been selected (e.g. a satellite to paste a component, a central body to paste a ground station, etc.). If the selected entity is not a valid destination for the entity currently in the clipboard, the paste operation will fail. If the destination is the same entity as the one currently in the clipboard (i.e. when hitting Ctrl+C and Ctrl+V in sequence), a duplicate of the selected entity is created on the same level of the hierarchy. Note that in order to paste a component as a sub-component of itself, a special paste entry is available in the context menu as Paste as subpart.

When copy-pasting an entity, all of its structural properties (i.e. those set in the Structure tab) are copied and pasted, as well as all its scenario properties (i.e. those set per-application and per-state in the Scenario editor tab). Entities are pasted as they were upon copy: if an entity is copied, modified, then pasted, the new entity will have the properties its source entity had at the time of the copy.

The following table lists the compatible destination entities for each source entity:

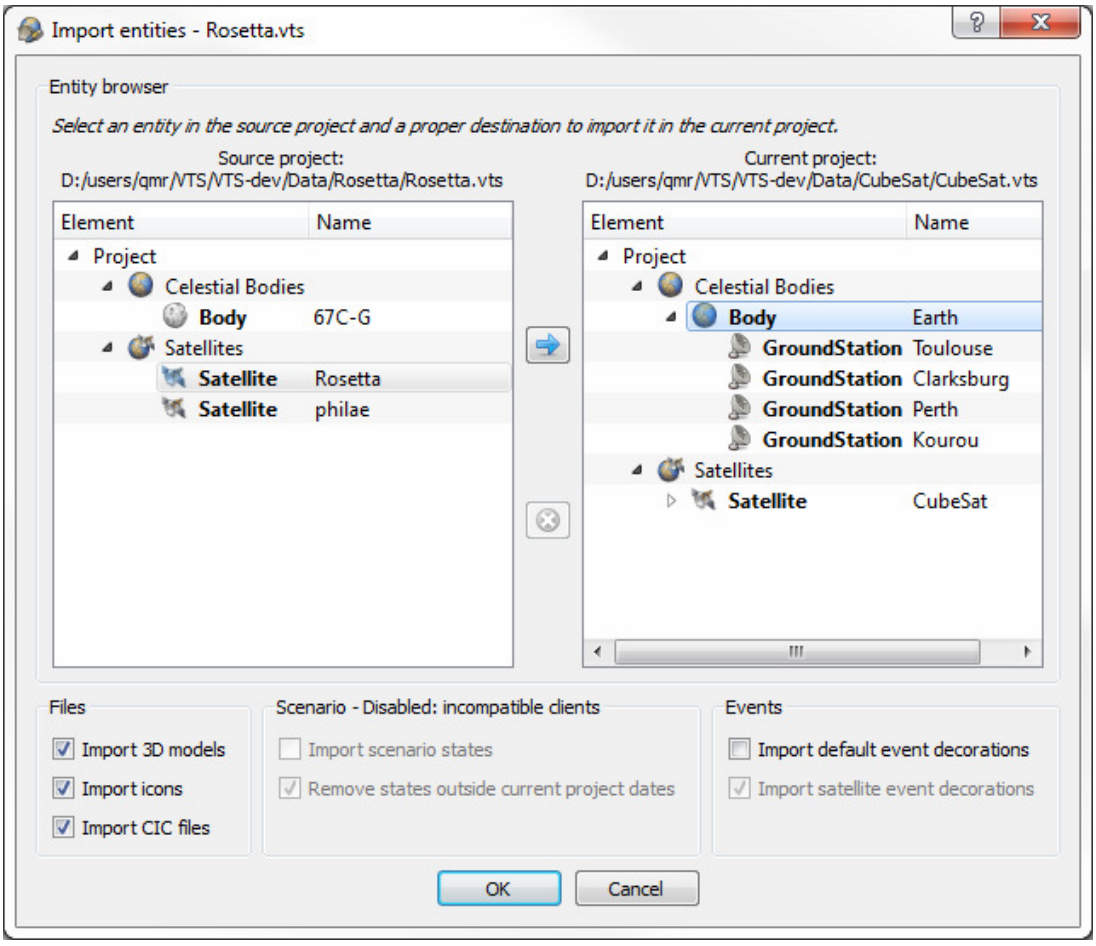
Source entity type	Valid destinations
Central body	**Project**, **Celestial bodies** item
Ground station	**Body**, **Ground stations** item
Point Of Interest	**Body**, **Points Of Interest** item
Region Of Interest	**Body**, **Regions Of Interest** item
Satellite	**Project**, **Satellites** item, **Body**
Component	**Satellite**, **SubPart**
Sensor	**Satellite**

Destinations for pasting a copied entity

6.1.3.8. Importing entities from an external project

VTS allows entities to be imported into the current project, from an external project file. This feature is available

through the Import entities entry in the Project menu.



Dialog for importing entities from an external project

The hierarchy of the source project is displayed in the left pane, while the hierarchy of the current project is displayed in the right pane.

To import an entity from the source project, simply select the entity to be imported in the source hierarchy, its destination entity in the destination hierarchy, and click the import arrow button located between the two panes. Valid destinations are listed in the above table (in the #Copy-pasting entities section section). The import of an entity can be canceled by selected it in the right pane and clicking the cross button located between the two panes.

The Files group allows enabling or disabling the following options:

- **Import 3D model**: import the 3D models and textures used by imported entities
- **Import CIC files**: import the CIC/CCSDS data files used by imported entities

The relative paths to the imported files in the source project will kept after the import. If a filename collision occurs, the user will have to decide whether to skip the import of the conflicting file, rename it, or overwrite the existing file.

The Scenario group is only enabled when client applications are identical in both the source and destination projects. It allows enabling or disabling the following options:

- **Import scenario states**: import the scenario states from the source project which do not already exist in the destination project
- **Remove states outside current project dates**: do not import scenario states from the source project which are outside the date range of the destination project

The Events allows enabling or disabling the following options:

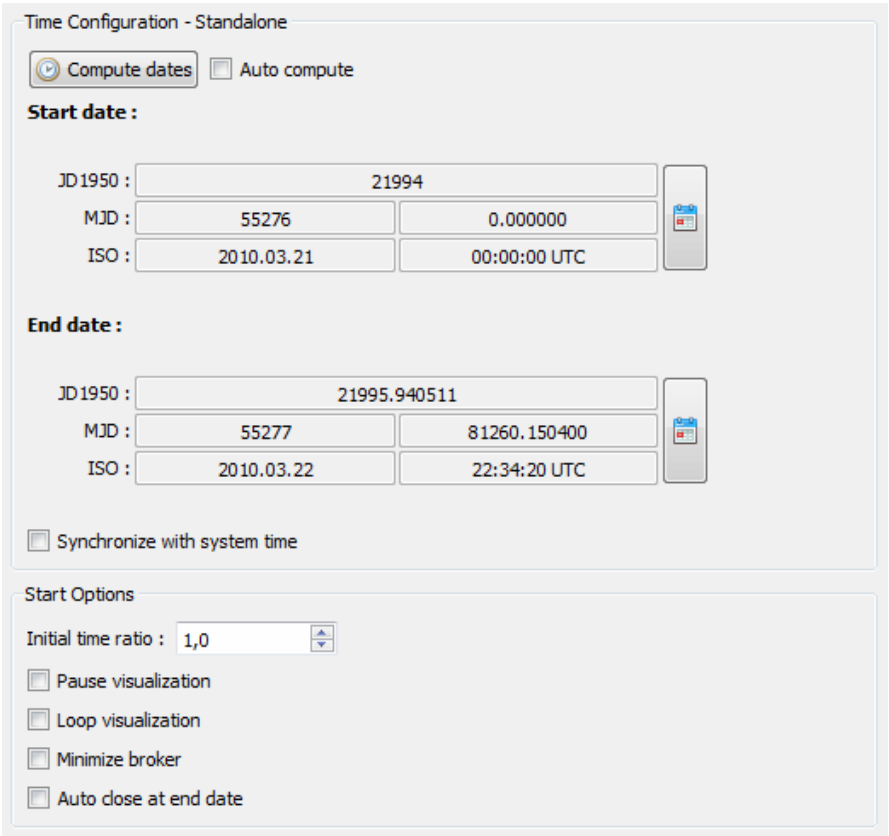
- **Import default event decorations**: import the default event decorations for all satellites
- **Import satellite event decorations**: import event decorations for the imported satellites (only available when at least one satellite has been selected for import)

Once the import has been configured, it can be applied by clicking the OK button, or canceled by clicking the Cancel button. Once the import has been applied, it can still be undone thanks to the Undo feature of VTS. Beware that file operations will not be undone.

6.1.4. Configuring a project and its entities

6.1.4.1. General parameters of a project

The general parameters of a project are its start and end dates, and the initial properties of the visualization (used by the Broker).



Configuration of the project dates and initial properties of the visualization

6.1.4.1.1. Configuring the project dates

- The **Compute dates** button automatically computes the date range of the project based on the widest common time interval of all ephemerides files used in the project. Note: dates are not updated when a data file or entity is added/removed from the project.

- The **Auto compute** option enables automatic computation of the project's date range when the user starts the visualization. This option is not saved in the project XML and can be changed manually after loading the project if needed. By default this option is unchecked, but this can be modified in the **Options/Settings...** dialog.
- The **Change date...** button located besides the project start and end dates allows manually editing the dates in CNES julian day format (JD1950), modified julian day format (MJD), and calendar format (ISO).
- The **Synchronize with system time** option allows synchronizing the visualization time with the computer's system clock. In this mode, time controls are disabled during visualization. Data files must provide data for the visualization time range.

Refer to the Date formats in VTS for further information on dates.

6.1.4.1.2. Configuring initial properties of the visualization

- The **Initial time ratio** setting allows specifying the initial visualization time ratio
- The **Pause visualization** option allows starting the visualization paused/unpaused
- The **Loop visualization** option allows the visualization to start over once it reaches the end of the time range
- The **Minimize broker** option allows the Broker to start minimized in the taskbar
- The **Auto close at end date** option allows the Broker to automatically close once the visualization reaches the end of the time range

6.1.4.2. Configuring an entity

Several entities in the project rely on some common parameters, which are described in the current section. Parameters specific to an entity are described in the sections concerning their respective entities.

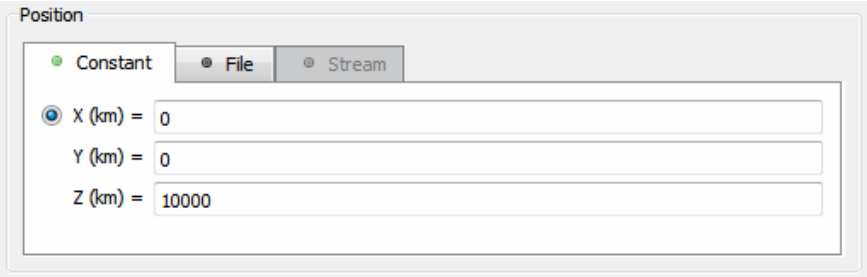
6.1.4.2.1. Position and orientation of an entity

This section describes the common mechanisms used for defining the position and orientation of an entity. Details specific to the various entities (such as the units used) are described in their respective sections.

For further information on how position and orientation are defined, refer to the Position of objects in VTS and Orientation of objects in VTS sections.

For further information on the available data sources, refer to the Data sources in VTS section.

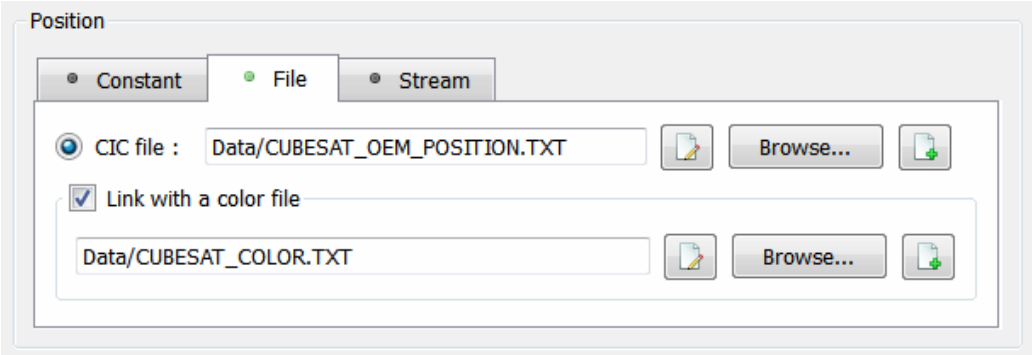
6.1.4.2.1.1. Fixed position



Fixed position

- The radio button in the **Constant** tab selects a fixed position for the current entity. Note that in the EME2000 reference frame, a fixed position is usually meaningless. This position mode is mainly used for defining the coordinates of a component in the satellite's local frame.

6.1.4.2.1.2. Position from a file



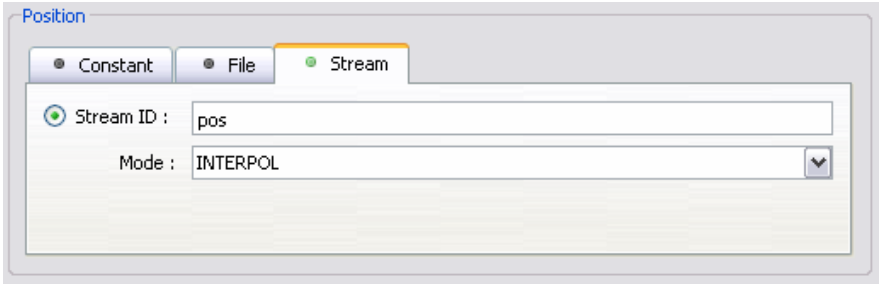
Position defined by an ephemerides file

- The radio button in the **File** tab allows specifying the path to a CIC/CCSDS file containing the position ephemerides for the current entity. The CIC/CCSDS file format is defined in the CIC/CCSDS data files in VTS section.
- The **Edit file...** button opens the currently selected file in a text editor.
- The **Browse...** button opens a dialog to select an existing CIC/CCSDS file from disk. If the selected file is not located in a sub-folder of the project folder, a dialog offers to copy it inside the project folder.
- The **New CIC file...** button creates an empty CIC/CCSDS file for the user to edit manually. Note that the CIC/CCSDS file header will depend on the currently selected type of position or orientation.
- The **Link with a color file** option allows specifying a CIC/CCSDS color file associated with the position file, through the same interface as for the position file. The color file will be used to colorize the satellite's orbit path in the 2D/3D views.

Color files format must meet the following requirements:

- MEM format with 3 (or 4) real fields ranging from 0 to 1, for the RGB components (and transparency)
- MEM format with 3 (or 4) integer fields ranging from 0 to 255, for the RGB components (and transparency)
- MEM format with 1 string field in HTML color format ("#RRGGBB" in hexadecimal), for the RGB components (no transparency)

6.1.4.2.1.3. Position as a stream

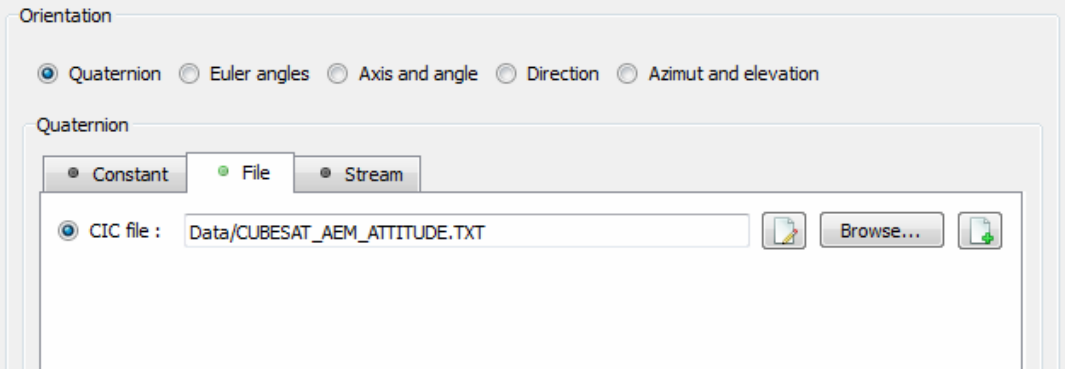


Position defined by a stream

- The radio button in the **Stream** tab allows specifying the ID of a data stream.
- The stream mode can be selected as *INTERPOL* or *DIRECT*. In *INTERPOL* mode, data will be interpolated, while in *DIRECT* mode data will be used as is, without interpolation. Note that the *INTERPOL* mode induces a delay in the visualisation.

Refer to the Real-time VTS section in the Synchronization protocol for VTS clients chapter for further information.

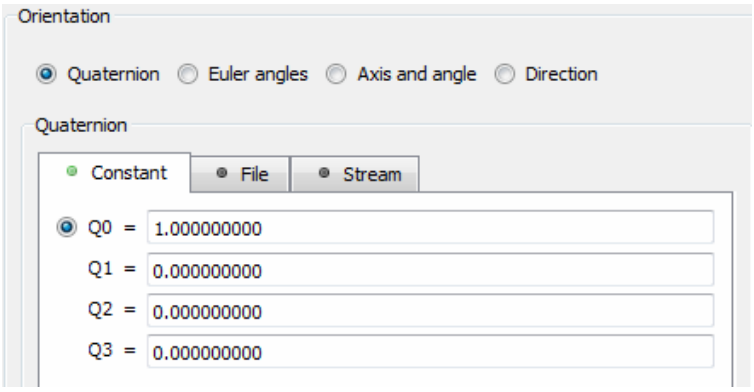
6.1.4.2.2. Orientation modes



Orientation defined by an ephemerides file

- Five radio buttons allow selecting the orientation mode of an entity: **Quaternion**, **Euler angles**, **Axis and angle**, **Direction**, **Azimuth and elevation**. Each of these orientation modes can be configured to use a fixed value, a sampled value in a CIC/CCSDS data file, or a value stream (as described above for the configuration of the position).

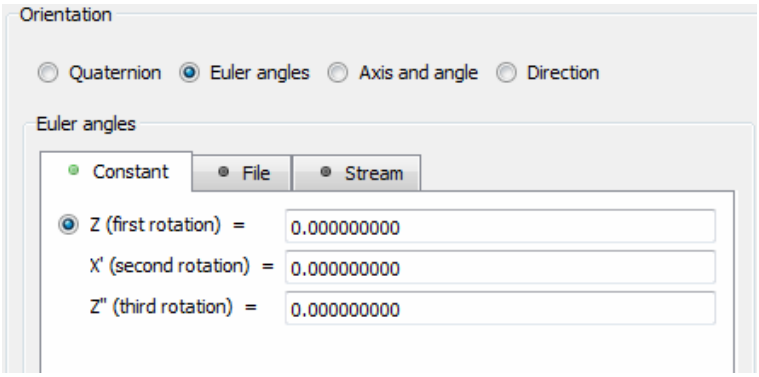
6.1.4.2.2.1. Orientation by quaternion



Orientation by constant quaternion

- An orientation by quaternion is defined by its four components: **Q0**, **Q1**, **Q2** and **Q3**.

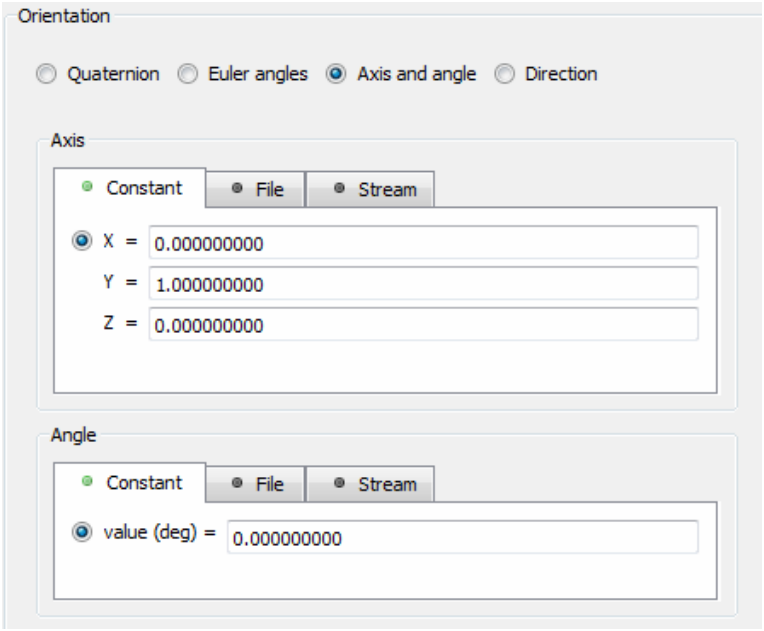
6.1.4.2.2.2. Orientation by Euler angles



Orientation by Euler angles

- An orientation by Euler angles is defined by a sequence of three rotations as follows: **Z**, **X'**, **Z''**

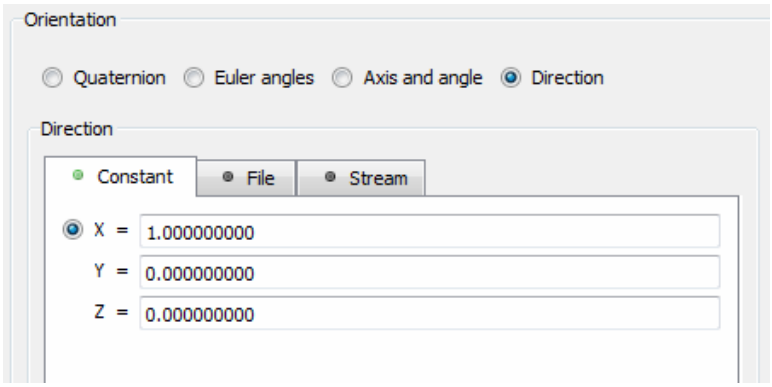
6.1.4.2.2.3. Orientation by axis and angle



Orientation by axis and angle

- An orientation by axis and angle is defined by the direction vector of the axis and the angle of rotation (in degrees). Both can be defined independently.
- It is possible to define a fixed axis and a sampled/streamed angle, as well as a sampled/streamed axis and a fixed angle. However, it is not possible to define both a sampled/streamed axis and a sampled/streamed angle.

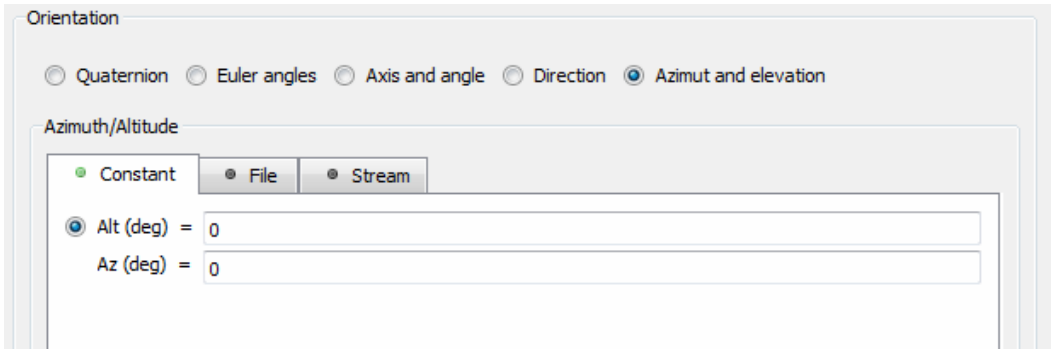
6.1.4.2.2.4. Orientation by direction



Orientation by direction

- An orientation by direction is defined by a direction vector in the entity's local frame. **This orientation mode does not strictly define an orientation**, there remains a degree of freedom around the direction of the vector. This mode offers a simple solution to define the orientation of a satellite component based for example on a file containing the direction of the Sun in the satellite's local frame.

6.1.4.2.2.5. Orientation by azimuth and elevation



Orientation by azimuth and elevation

- An orientation by azimuth and elevation is defined by two angles for the azimuth and elevation, both in degrees. **This orientation mode does not strictly define an orientation**, there remains a degree of freedom around the direction of aim.

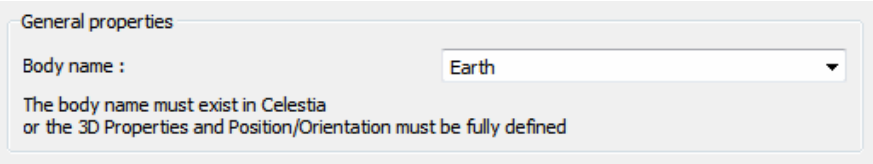
6.1.4.3. Configuring a central body

Missions visualized in VTS are defined around central bodies. A spacecraft is attached to a central body, and a central body is attached to the Sun. At least one central body must be defined for the VTS project to be valid. New projects use the Earth as default central body.

6.1.4.3.1. General properties of a body

The general properties of a body are its name and texture.

6.1.4.3.1.1. Name of a body



General properties of a body

- If the central body has a valid Celestia body name, its default ephemerides can be used or overridden (refer to the #Position and orientation of a body section below). A drop-down list allows choosing a body name amongst the planets of the solar system. The planet's radius is automatically filled in in the 3D properties of the body.
- A body not known to Celestia can be defined. However, it will then be necessary to provide user-defined 3D properties and ephemerides.

6.1.4.3.1.2. Texture of a body

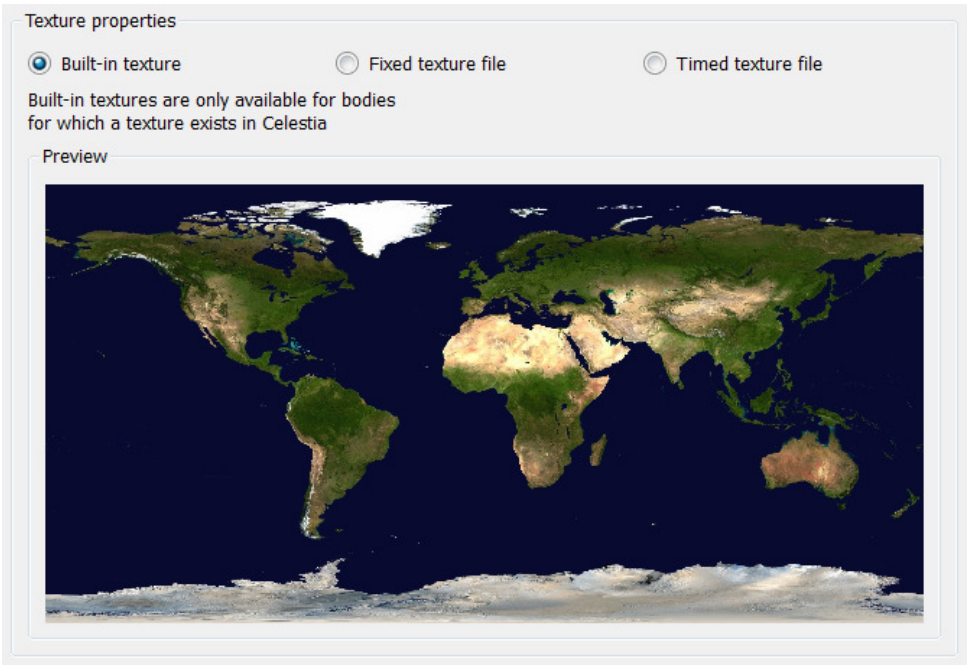
The texture of a body defines its appearance in the 2D and 3D views. It can be defined in 3 ways, selected by a radio button: Built-in, Fixed, Timed.

The Preview area displays the currently selected texture for the current body.

Notes:

- User-defined textures must correspond to a plate carrée projection, i.e. with a 1:2 ratio and centered on longitude 0°.
- Due to limitations in Celestia, only one central body may used the *Fixed* or *Timed* texture types in the project.

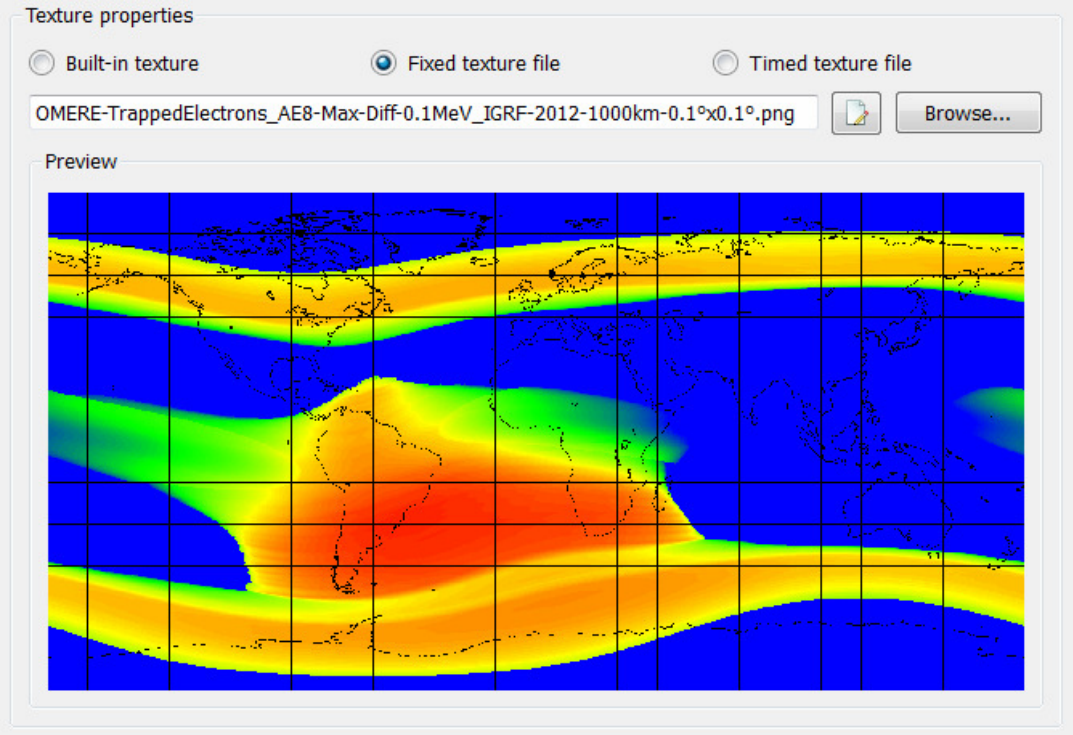
Built-in texture



Built-in texture

The texture is provided by Celestia, if available. Celestia has built-in textures for all planets of the solar system and some of their natural satellites.

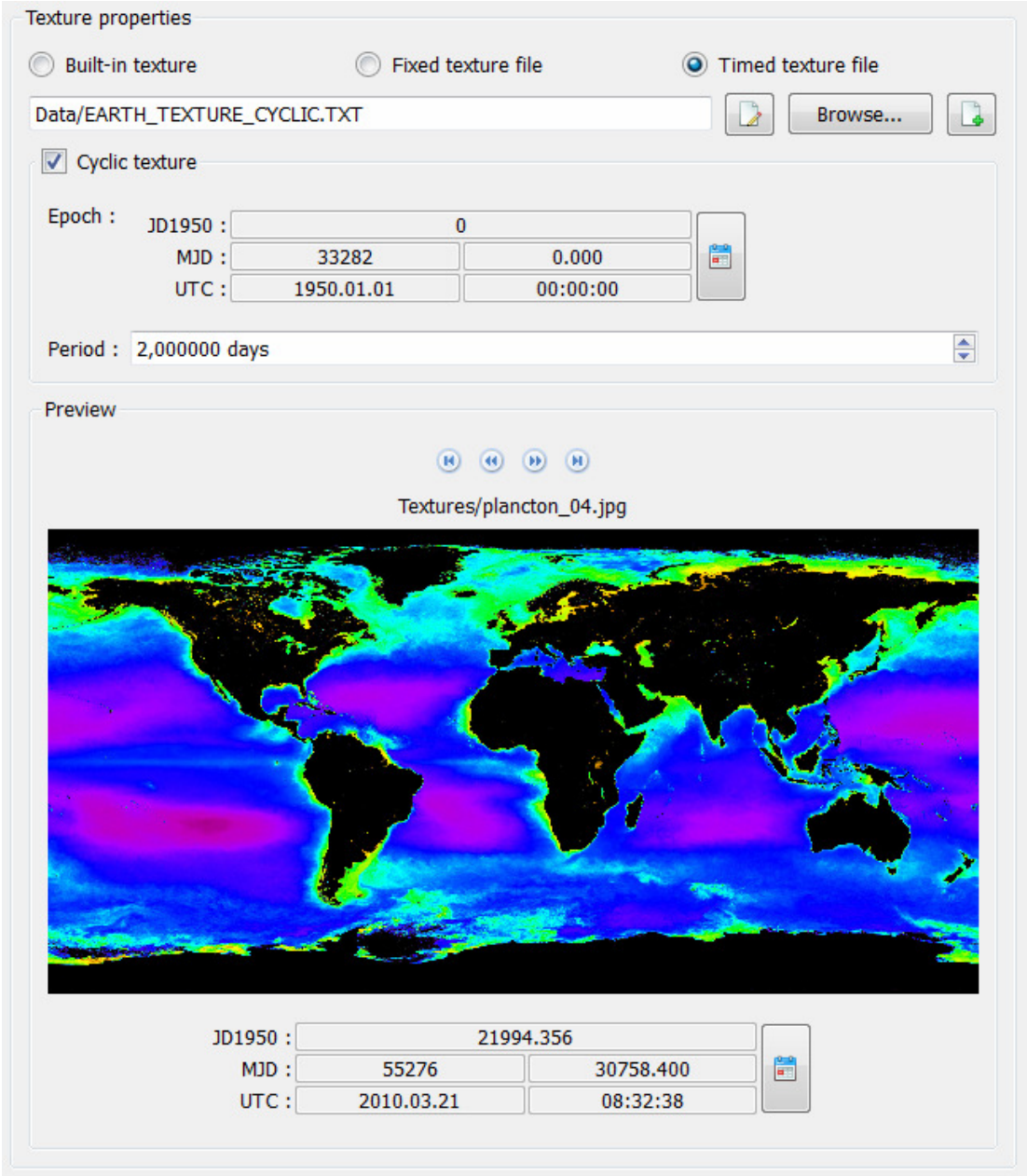
Fixed texture



Fixed texture

The texture is defined by an image file in the project folder.

Timed texture



Timed texture

The texture is defined through a CIC/CCSDS file in the MEM format. This file contains the relative path (in the project folder) to the texture to use with regards to the current visualization date.

Several parameters must be defined for a timed texture:

- The MEM timed texture file, containing *STRING* data of dimension *1*
- The cyclic/acyclic property of the texture

Notes:

- The *Preview* area allows navigating through the textures of the MEM files, either with the navigation buttons, or by directly specifying a date

- If a line of the MEM file contains the string *DEFAULT*, the built-in texture of the body is used for this date

Beware, using a timed texture can make the visualization jerky, especially when using Celestia. The following circumstances may cause performance issues:

- timed texture with frequent texture changes (every few seconds of wall-clock time)
- timed texture with textures of significant sizes
- timed texture visualized at high time ratio (causing texture changes every few seconds of wall-clock time)

Acyclic timed texture

For an acyclic texture, dates in the MEM file are the dates at which the texture changes occur. For example, the following line:

```
56018 0 "Textures/Chl_a_april 2012.png"
```

specifies that image file Textures/Chl_a_april 2012.png is to be used as texture for the central body at MJD date 56018 0.

Cyclic timed texture

For a cyclic texture, dates in the MEM file describe the instants of the cycle at which texture changes occur. Two additional parameters are required:

- The epoch of the cycle
- The period of the cycle in fractional days

The epoch of the cycle must be prior to the start date of the project. Periods of an variable number of days, such as a month or a year, may not be specified. The best solution in this case is to use an acyclic timed texture with texture changes adjusted for the project.

Dates in the MEM file are interpreted as durations from the start of the cycle. This means that a timed texture MEM file must contain dates in the MJD date format. For example, the following lines:

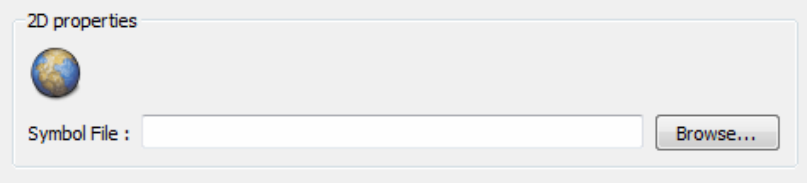
```
0     0 "Textures/plankton_00.jpg"
0  7200 "Textures/plankton_01.jpg"
```

specify that the image file Textures/plankton_00.jpg is to be used as texture at the start of a cycle, and that the image file Textures/plankton_01.jpg is to be used as texture two hours after the start of a cycle.

Beware, if MJD date 0 0 is not present in the MEM file, no texture will be displayed at the start of a cycle.

6.1.4.3.2. 2D properties of a body

The 2D properties of a body allow configuring its appearance in the project hierarchy.

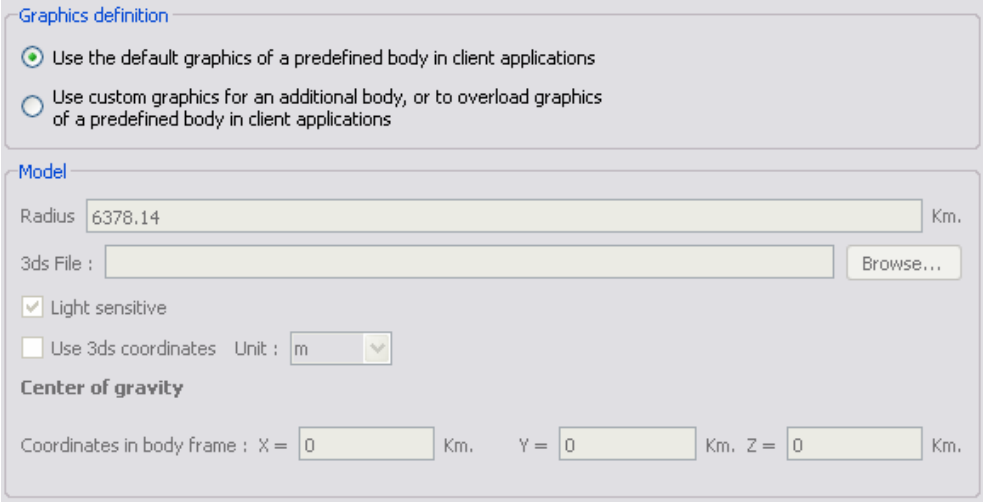


Default icon for a body

- If the **Symbol file** field is empty, a default icon is assigned to the body.

6.1.4.3.3. 3D properties of a body

The 3D properties of a body define its appearance in 3D views.



3D properties of a body

The Graphics Definition radio buttons select the 3D properties to use for the body:

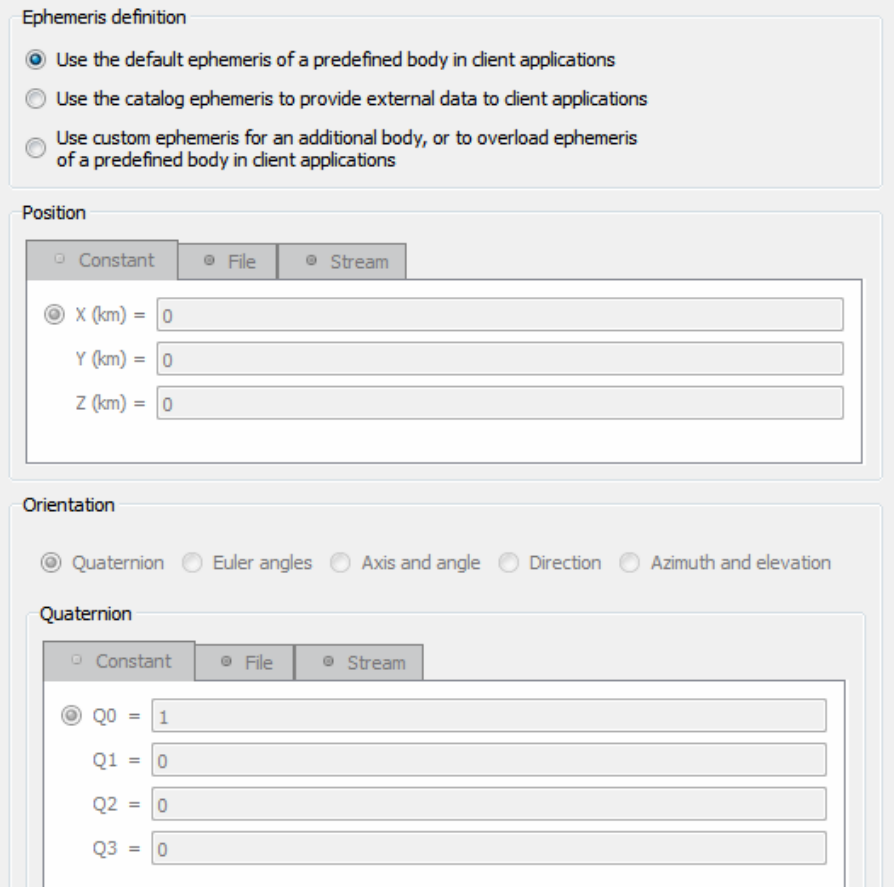
- default: use the graphics properties available if Celestia, if any.
- custom: redefine the 3D mesh of an existing Celestia body or define the 3D mesh of a body not available in Celestia. The parameters are similar to those described in the 3D properties of a satellite section. Values are expressed in kilometers.

All custom 3D properties field must be filled in when redefining the 3D mesh of an existing Celestia body.

Custom 3D properties must be selected and a 3D mesh must be defined if the current body is not available in Celestia.

6.1.4.3.4. Position and orientation of a body

The VTS toolkit is mainly geared towards the visualization of missions around central bodies of the solar system. In general, the ephemerides of these bodies are known. However it remains possible to override the ephemerides of predefined bodies, or to define the ephemerides of a non-default body.



Configuring the ephemeris mode

The Ephemeris Definition radio buttons select the position and orientation to use for the body:

- Default ephemeris mode : client applications choose their preferred ephemeris origin
- Catalog ephemeris mode : client applications use VTS catalog ephemeris files
- Custom ephemeris mode : the user provides ephemeris to the body as described in the Position and orientation of an entity section.

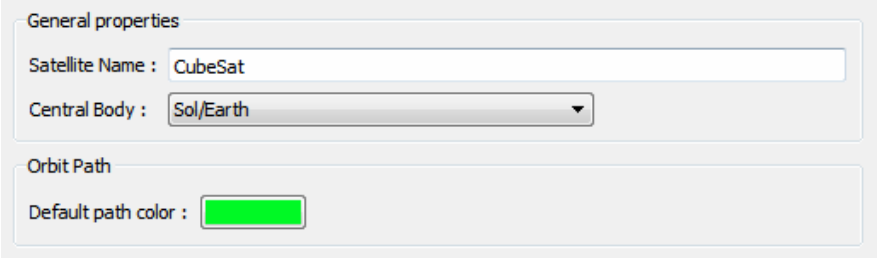
See the Central bodies in VTS section for more details.

6.1.4.4. Configuring a satellite

A satellite is composed of a main component and sub-components. General, 3D, and geometric properties are configured at the component level. Common and 2D properties are configured only for the main component.

6.1.4.4.1. General properties of a satellite

The general properties of a satellite are its name, central body, and parameters of its orbit path.



General properties of a satellite

6.1.4.4.1.1. Name and central body of a satellite

The name of the satellite will be the satellite's unique identifier during visualization, used to identify the satellite in view properties and messages between the Broker and client applications.

The central body of the satellite defines the reference frame for the satellite, in which the satellite's position is expressed.

6.1.4.4.1.2. Orbit path of a satellite

The orbit path of a satellite is displayed in the 2D and 3D views. It has the following property:

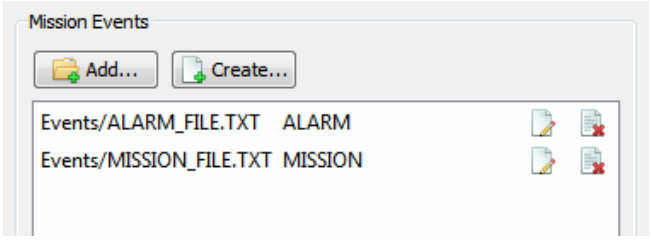
- The **Path color** field defines the display color of the orbit path. This color is used unless a color file is specified along with the sampled position of the satellite. Refer to the Position from a file section for further information.

Note: the displayed length of the satellite's orbit path is now configured in the project scenario.

6.1.4.4.2. Events attached to a satellite

Mission events can be attached to the satellite to be displayed in the project timeline and in compatible client applications. These events are provided in CIC/CCSDS files in MEM format.

- The **Add...** button allows selecting an additional CIC/CCSDS event file
- The **Create...** button allows creating an empty CIC/CCSDS event file
- The **Edit file...** button on each line opens the selected CIC/CCSDS event file in a text editor
- The **Remove file** button on each line removes the selected CIC/CCSDS file from the list



List of mission event files

Each file has its top-level event type displayed next to its name on its line. This event type is used to construct the full event type name for events of the file.

The decoration of all events can be controlled in the Event Type Editor tab of the VTS configuration utility (see the Configuring event types section). The visibility of the events can be controlled during visualization through the Broker (see the Events tab section in the Broker user manual).

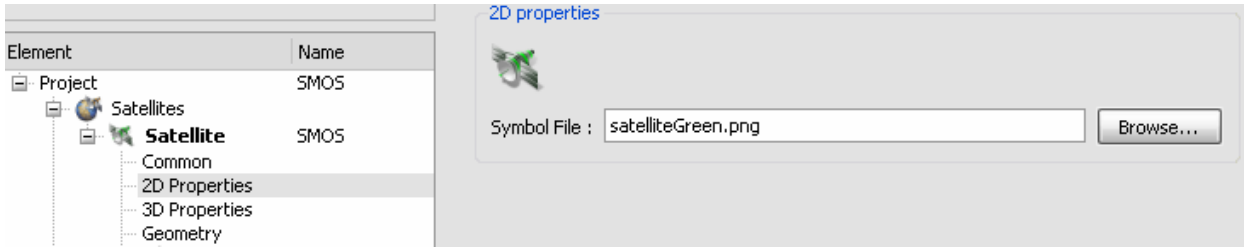
Refer to the Mission events in VTS chapter for more information on events in VTS.

6.1.4.4.3. 2D properties of a satellite

The 2D properties of a satellite define the appearance of the satellite in the 2D views and in the project hierarchy.

Default icon for a satellite

- If the **Symbol file** field is empty, a default icon is assigned to the satellite.

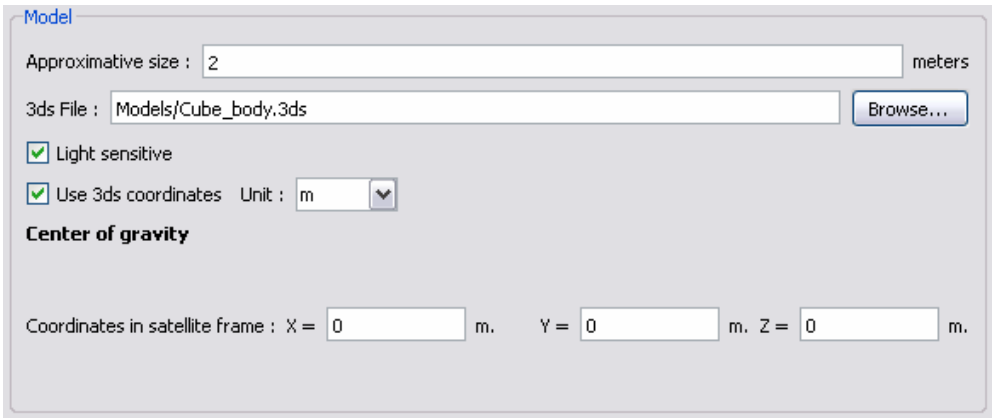


Custom icon for a satellite

- The user may select an image file to redefine the appearance of the satellite in the 2D view and project hierarchy. If the selected file is not located in a sub-folder of the project folder, a dialog will offer to copy the file inside the project folder.

6.1.4.4.4. 3D properties of a satellite

The 3D properties of a satellite define its appearance in 3D views.



3D properties of a satellite

- The **Approximate size** field only applies to the current component (3ds file). It has two meanings:
  - If the **Use 3ds coordinates** option is enabled, it defines the bounding sphere of the 3D mesh to be used by clipping mechanisms. This parameter should be set to a realistic value.
  - If the **Use 3ds coordinates** option is disabled, it defines the actual dimensions to be applied to the 3D mesh of the component.

- The **3ds File** field defines the 3ds mesh file to use for the component. It must be point to an AutoDesk 3DS Max file. If the selected file is not located in a sub-folder of the project folder, a dialog will offer to copy it inside the project folder.

- The **Light sensitive** option enables or disables shading of the 3D mesh. Objects such as satellite bodies or solar arrays usually should be shaded, while abstract objects such as frame axes or vectors should not. Unshaded objects are only lighted in a diffuse manner, and do not show specular reflections from light sources such as the Sun.

- The **Use 3ds coordinates** option allows using the intrinsic coordinates of the 3ds mesh file. If the 3ds mesh file originates from a known CAD source, it is advised to leave the option enabled and to set the **Approximate size** field accordingly to define a bounding sphere for the 3D mesh. If the 3ds mesh file originates from an unknown source, it is advised to disable the option and to set the **Approximate size** field to the desired size for the 3D mesh. The **Unit** drop-down list defines the unit to use when reading coordinates in the 3ds mesh file.

- The **Center of gravity** (for a satellite) or **Rotation center** (for a subpart) field defines the XYZ coordinates of this point in the current item's local frame, in meters.

Note: Modifying the center of gravity of a satellite has no effect on the origin of the satellite's frame. The satellite's local frame can only be altered by directly modifying the 3D mesh using specialized 3D software.

6.1.4.4.5. Position and orientation of a satellite

The position and orientation of a satellite can be configured as described in the Position and orientation of an entity section. They are expressed in the EME2000 reference frame (inertial equatorial earth-centered). The unit for position values is the kilometer.

6.1.4.5. Configuring a sub-component

A sub-component is a part of a satellite detached from the main component. A satellite is composed of a main component, and may be composed of any number of sub-components. A sub-component itself may be composed of any number of sub-components. Sub-components can be animated in the local frame of their parent component. The definition of a sub-component is quite similar to the definition of the main component of a satellite.

6.1.4.5.1. General properties of a sub-component

The general properties of a component define its name in the project hierarchy. This name will be used as a unique identifier during visualization, by commands interacting with entities.

6.1.4.5.2. 3D properties of a sub-component

The 3D properties of a sub-component define its appearance in 3D views. They are similar to the 3D properties of a satellite. Refer to the 3D properties of a satellite for further information.

- The coordinates of the rotation center are expressed in the satellite's frame, in meters.

6.1.4.5.3. Position and orientation of a sub-component

The position and orientation of a sub-component are defined as described in the Position and orientation of an entity section. The transformations are expressed in the parent component's frame. For example, for a solar array split in several levels of components, the coordinates of the first component are expressed in the satellite's frame, those of the second component are expressed in the first component's frame, etc.

The unit for position values is the meter.

6.1.4.6. Configuring a satellite sensor

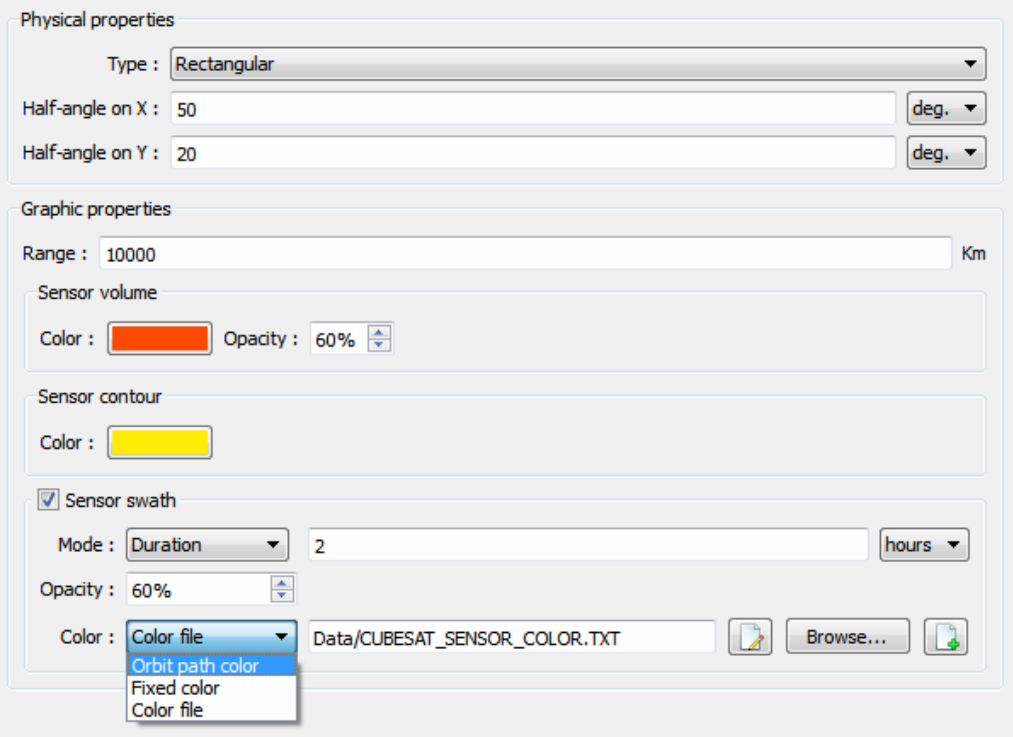
Satellite sensors can be attached to the main component of a satellite.

6.1.4.6.1. General properties of a satellite sensor

The general properties of a satellite sensor define the name of the sensor in the project hierarchy. This name will be used as a unique identifier during visualization, by commands interacting with entities.

6.1.4.6.2. Properties of a satellite sensor

By convention, in its canonical position a satellite sensor frame is aligned with the satellite's frame and the sensor points along the Z axis.



Properties of a satellite sensor

6.1.4.6.2.1. Physical properties of a satellite sensor

A sensor is described by the following physical properties:

- The sensor's shape: the **Type** drop-down list defines either an *Elliptical* (conical) or *Rectangular* (pyramidal) base for the sensor.
- The sensor's aperture: the **Half-angle on X** and **Half-angle on Y** fields define the half-angles of aperture around the X axis (in the YZ plane) and around the Y axis (in the XZ plane). These angles are expressed either in degrees or radians.

6.1.4.6.2.2. Graphic properties of a satellite sensor

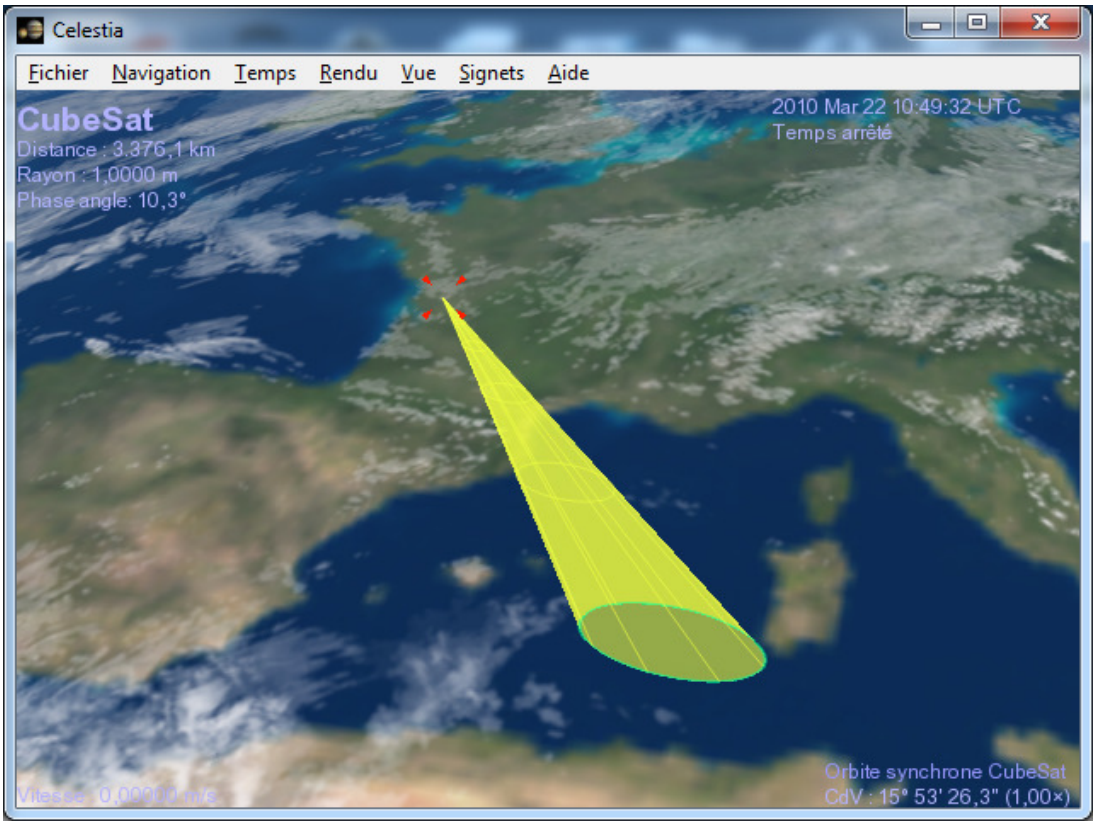
The graphical properties of a sensor define its appearance in the 2D and 3D views:

- The **Range** field defines the maximum length of the sensor volume in the 3D views, expressed in kilometers
- **Sensor volume** (3D views only)
    - The **Color** button defines the color of the sensor volume. This color is also used in the graphical interfaces to identify the sensor.
    - The **Opacity** field defines the opacity of the sensor volume (0: transparent, 100: opaque)
- **Sensor contour** (2D and 3D views)
    - The **Color** button defines the color of the sensor contour
- **Sensor swath** (2D and 3D views)
    - The **Mode** list defines the display mode of the sensor swath
        - The **Instantaneous** mode only displays the instant sensor swath (for the current location of the sensor; this corresponds to a trace of null duration)
        - The **Duration** mode allows defining the residual length of the sensor swath, expressed in hours or days
    - The **Opacity** field defines the opacity of the sensor swath (0: transparent, 100: opaque)
    - The **Color** list allows defining the sensor swath color using one of the following 3 methods:
        - The **Orbit path color** mode uses the same color as the one used for the satellite's orbit path (either defined in the general properties of the satellite, or by the CIC/CCSDS color file attached to the position file of the satellite)
        - The **Fixed color** mode uses a fixed user-defined color
        - The **Fixed file** mode uses the colors specified in a user-defined CIC/CCSDS color file

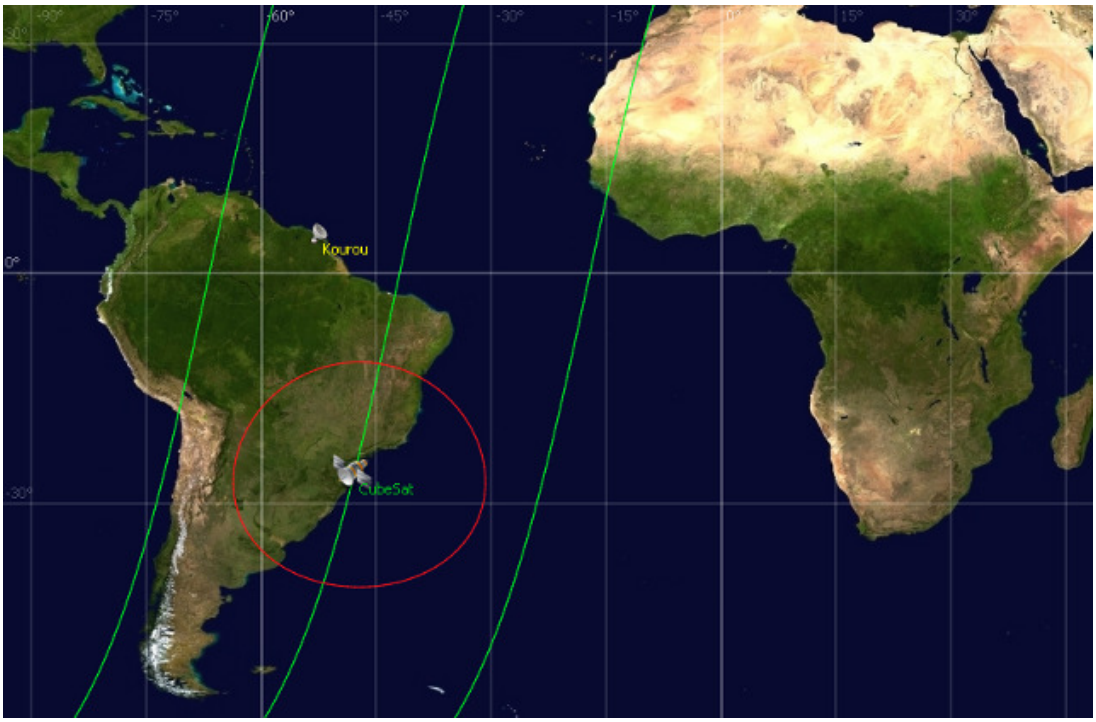
Color files format must meet the following requirements:

- MEM format with 3 (or 4) real fields ranging from 0 to 1, for the RGB components (and transparency)
- MEM format with 3 (or 4) integer fields ranging from 0 to 255, for the RGB components (and transparency)
- MEM format with 1 string field in HTML color format ("#RRGGBB" in hexadecimal), for the RGB components (no transparency)

Note: When using a color file, the black color (0 0 0) is used to disable rendering of the sensor swath while it is in effect. This allows drawing the sensor swath by "patches", e.g. to visualize instrument power-on/power-off cycles.



Satellite sensor in Celestia



Satellite sensor in 2DWin

6.1.4.6.3. Position and orientation of a satellite sensor

The position and orientation of a satellite sensor are defined as described in the Position and orientation of an entity section. Transformations are expressed in the satellite's frame.

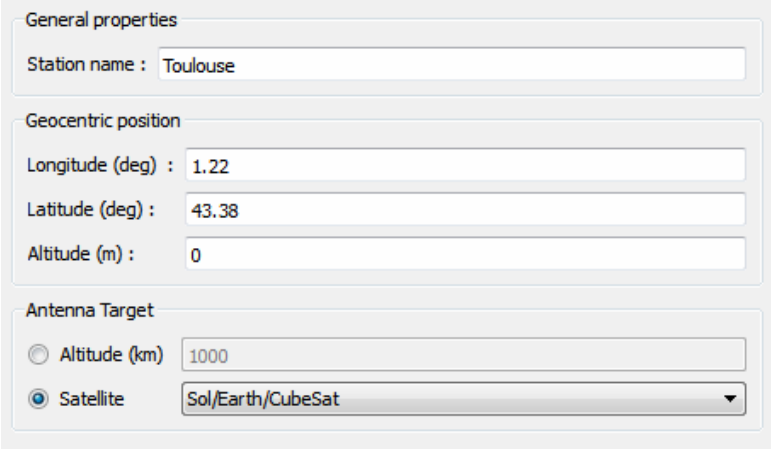
Note: The Direction and Azimuth and elevation modes are not available for the orientation of a sensor as the induced degree of freedom implies the orientation of the sensor cannot be correctly specified.

6.1.4.7. Configuring a ground station

Any number of ground stations can be attached to central bodies in the project.

6.1.4.7.1. General properties of a ground station

The general properties of a ground station define its name, position on its central body, and the target of its antenna.

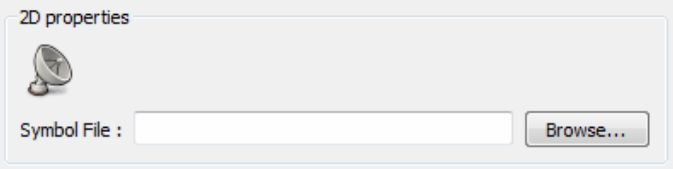


General properties of a ground station

- The name of a ground station must be unique on its parent body.
- The geographical coordinates (longitude and latitude, in degrees) of a ground station must be defined. Its altitude (in meters) is optional.
- The target of a ground station's antenna can be either a fixed altitude, or the altitude of one of the project's satellites.

6.1.4.7.2. 2D properties of a ground station

The 2D properties of a ground station define its appearance in the 2D views and in the project hierarchy.

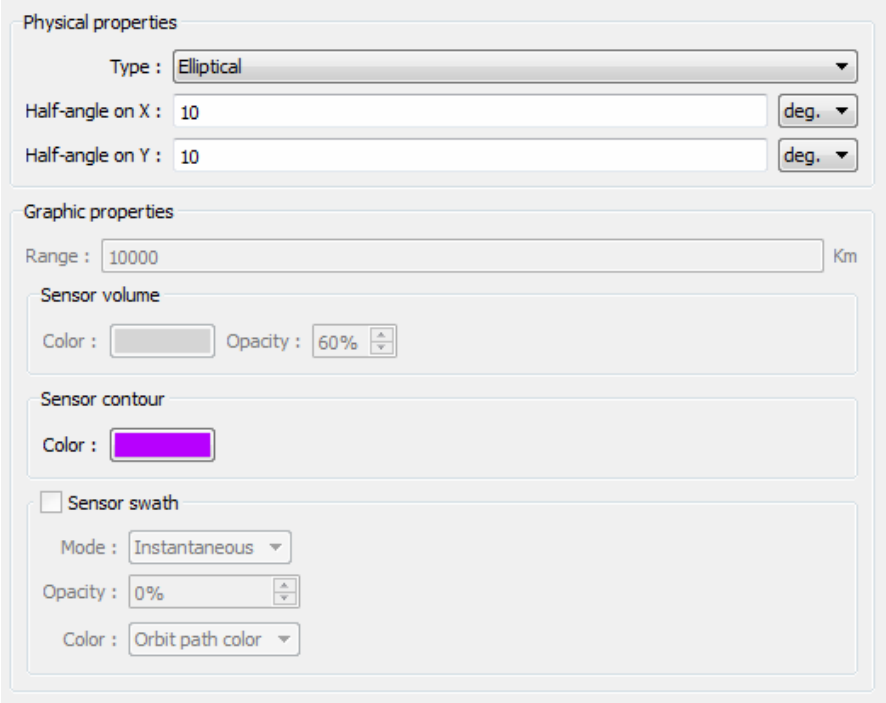


Default icon for a ground station

- If the **Symbol file** field is empty, a default icon is assigned to the ground station.

6.1.4.7.3. Properties of a ground station sensor

By convention, in its canonical position a ground station sensor frame is aligned with the ground station's frame and the sensor points along the Z axis.



Properties of a ground station sensor

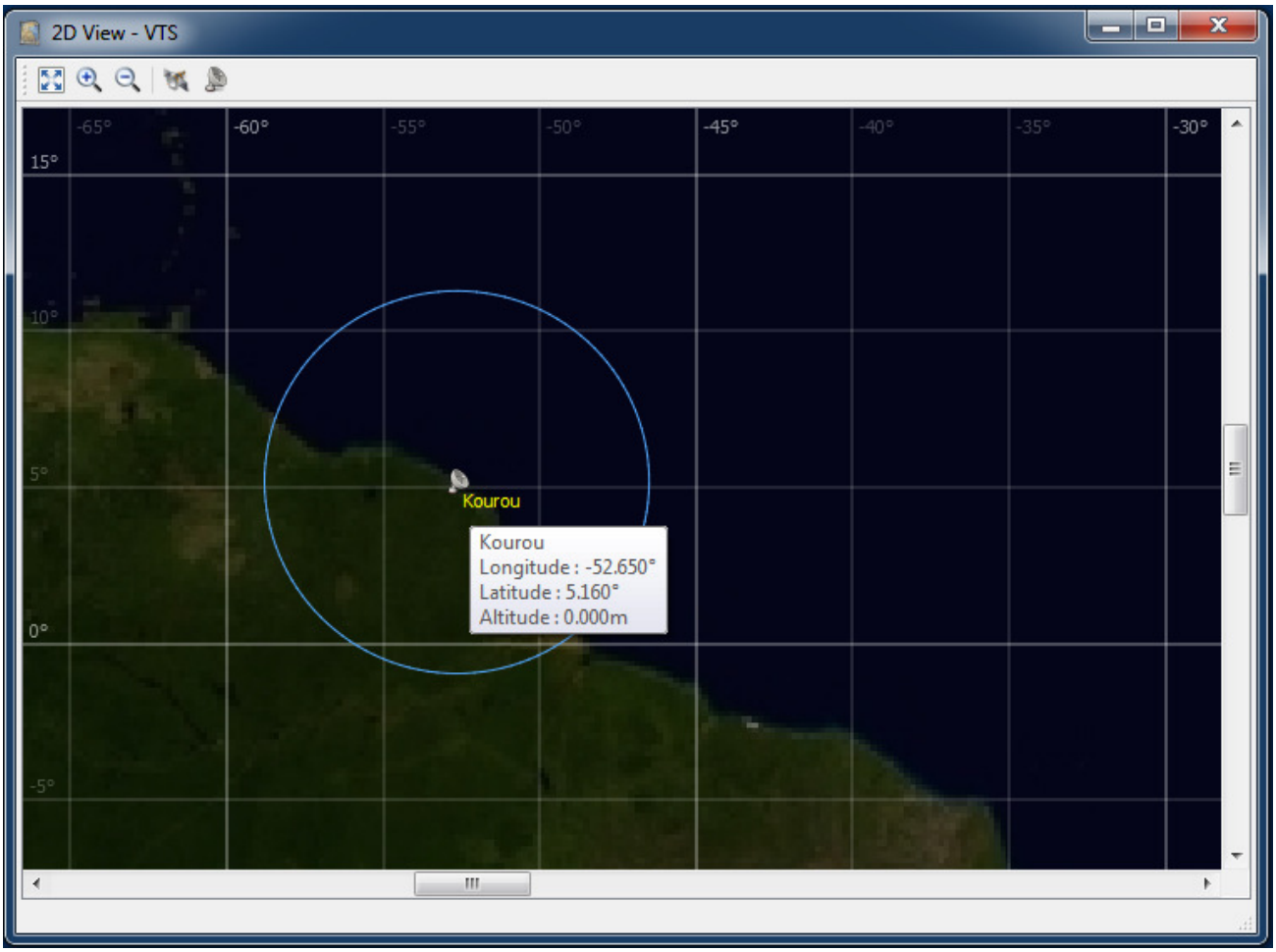
6.1.4.7.3.1. Physical properties of a ground station sensor

These properties are identical to those of a satellite sensor. Refer to the Physical properties of a satellite sensor section for further information.

6.1.4.7.3.2. Graphic properties of a ground station sensor

The graphic properties of a ground station sensor define the appearance of the sensor in the 2D views. Ground station sensors are not displayed in the 3D views, the corresponding parameters are henced grayed out.

- **Sensor contour**
  - The **Color** button defines the color of the sensor contour

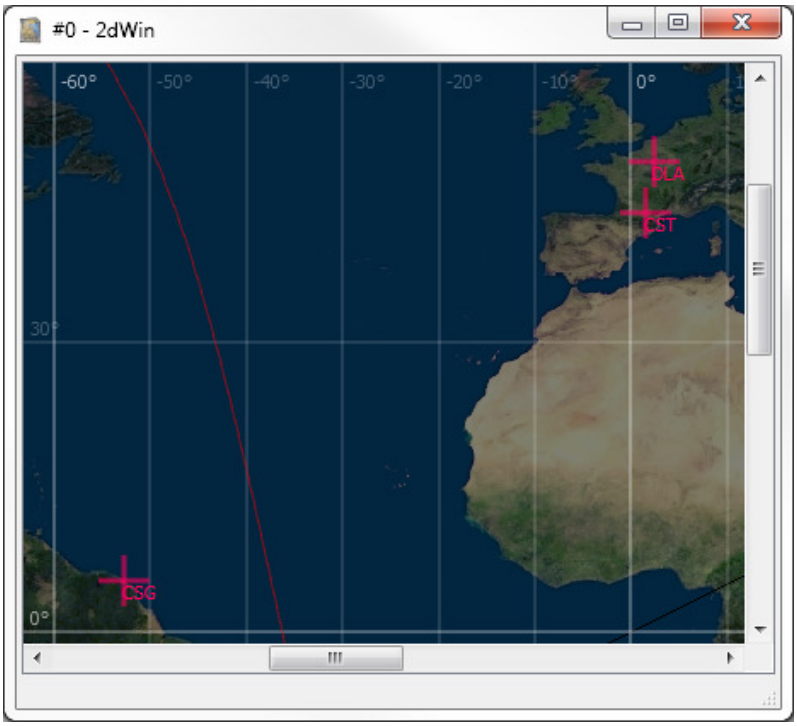


Ground station sensor in 2DWin

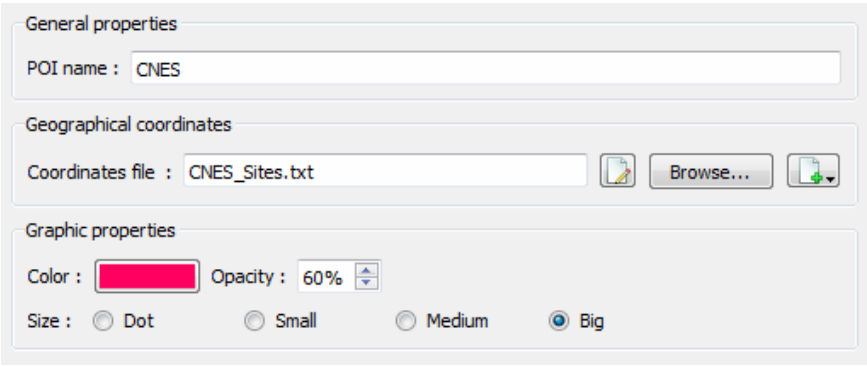
6.1.4.8. Configuring a Point Of Interest

A Point Of Interest (abbreviated POI) represents one or many locations in geographical coordinates. Any number of POIs can be attached to central bodies in the project.

Refer to the POIs and ROIs in VTS chapter for more information of the POI file format.



Points Of Interest in 2DWin



Properties of a Point Of Interest

6.1.4.8.1. General properties of a POI

The general properties of a POI define its name. It must be unique among other POIs in the same parent body.

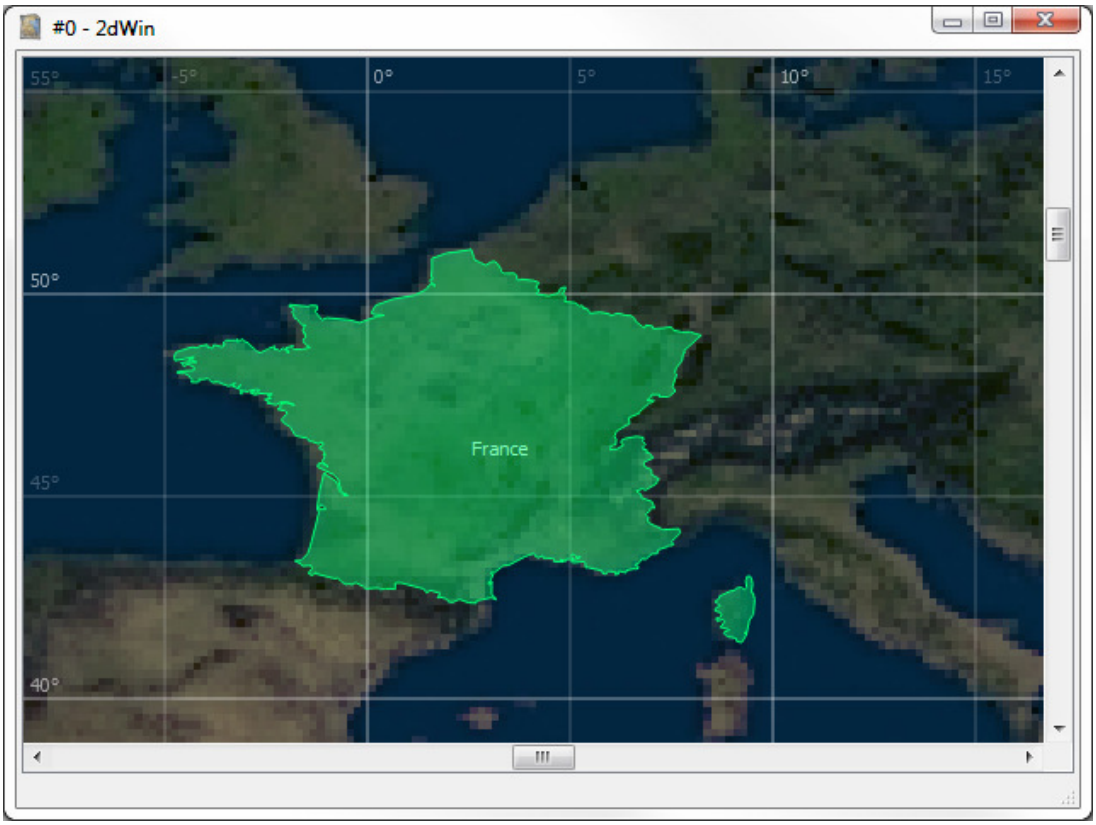
6.1.4.8.2. Graphic properties of a POI

The graphic properties of a POI define the shape of the location markers, as well as their color and opacity.

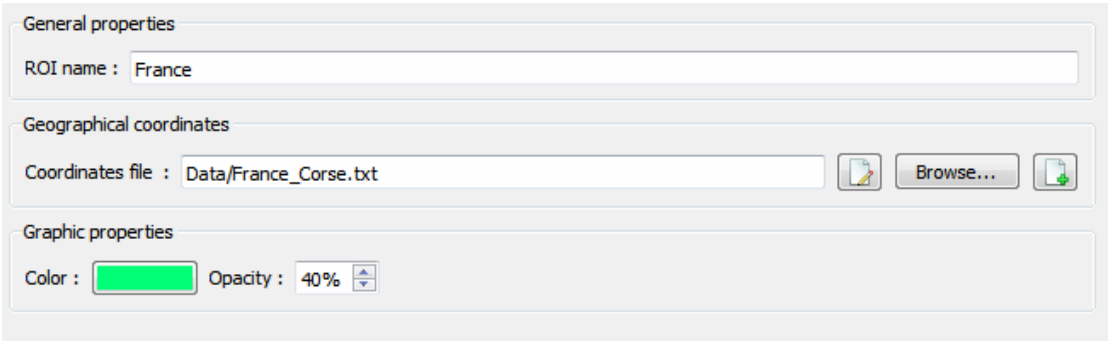
6.1.4.9. Configuring a Region Of Interest

A Region Of Interest (abbreviated ROI) represents one or many polygons in geographical coordinates. Any number of ROIs can be attached to central bodies in the project.

Refer to the POIs and ROIs in VTS chapter for more information of the ROI file format.



Region Of Interest in 2DWin



Properties of a Region Of Interest

6.1.4.9.1. General properties of a ROI

The general properties of a ROI define its name. It must be unique among other ROIs in the same parent body.

6.1.4.9.2. Graphic properties of a ROI

The graphic properties of a ROI define the color of the polygon and its opacity.

6.1.4.10. Configuring a client application

Any number of client applications may take part in the visualization of a project. The view properties of these applications can be configured in the project scenario.

Selecting an application in the project hierarchy some information and parameters:

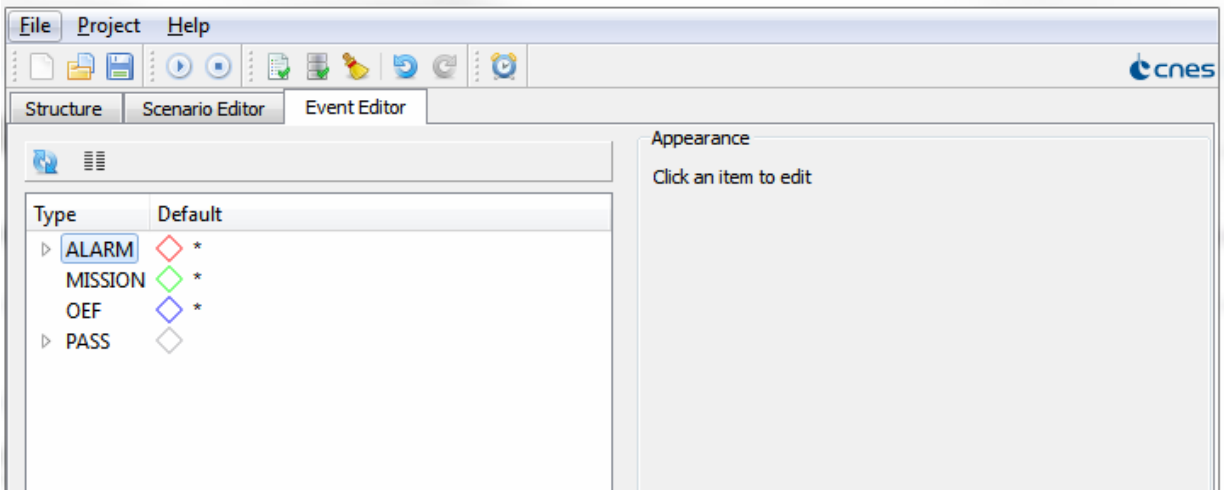
- Text information about the application (found in the application's *README* file).
- Some applications define parameters which must be passed on to the application once it has successfully connected to the visualization. Those can be specified in the **Initial States** area.
- Some applications require specific parameters to be passed on to their launchers. Those can be specified in the **Specific Args** field.

Configuration of an application

6.1.5. Configuring event types

Event types can be configured in the Event Type Editor tab of the VTS configuration utility. The appearance of the event types in the Timeline and 2D views can be specified for all events of the project. The GUI is composed of the following:

- An action toolbar
- A hierarchical view of all event types
- An editing pane for the currently selected event type



Event type editor

6.1.5.1. Loading event files

The event types displayed in the tree view are loaded from the event files attached to the project's satellites. Clicking the Reload Events button in the toolbar reads all event files and displays the event types available in these files. This should be done manually each time an event file is added to or removed from the project, or if existing event files are modified. Decorations of existing event types are not affected by a reload.

6.1.5.2. Configuring default event type decorations

When the Toggle Edit Entity Decorations Mode button is untoggled, the event type editor is in "basic" mode and only the All satellites column is visible. Decorations configured in this mode will apply to all satellites in the project.

The default decoration displayed in the tree for all types is a gray diamond. Clicking the decoration on the line of an event type selects it for edition in the editing pane. The Customized option can then be enabled to customize the event type's decoration:

- The **+** and **-** buttons add or remove decoration layers
- The arrow buttons move the selected layer up or down

The preview area displays the event type decoration as it will appear in client applications that handle events.

Stars in the hierarchical view indicate event types with custom decorations. Decorations are inherited by all descendant types of a decorated event type, unless one of these descendants itself has its decoration customized.

6.1.5.2.1. Shape layers

Shape decoration layers have the following properties:

- the shape of the layer (*Circle*, *Circular target*, *Cross*, *Diamond* or *Square*)
- the color of the layer
- whether or not to fill the shape

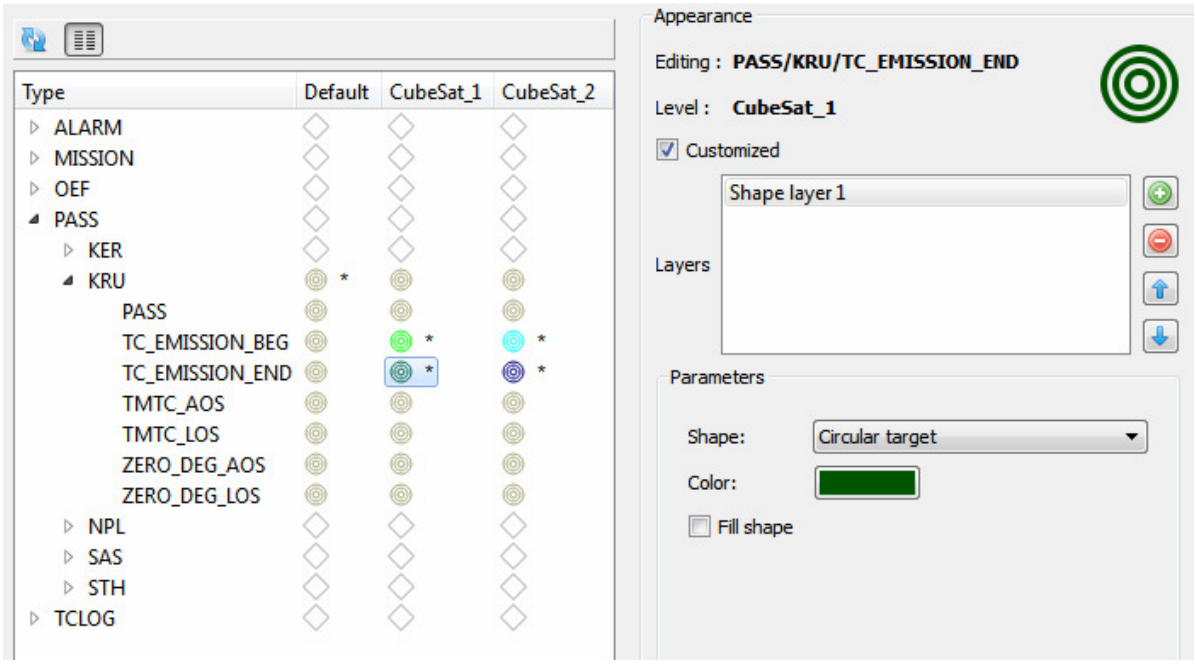
6.1.5.2.2. Icon layers

Icon decoration layers allow the user to select one of the following appearances:

- the entity's icon configured in the project
- an image/icon file located inside the project folder (preferred dimensions are 32x32, see the 2D icons and textures section of the Data description for VTS projects chapter for supported file formats)

6.1.5.3. Configuring satellite-specific event type decorations

When the Toggle Edit Entity Decorations Mode button is toggled, the event type editor switches to "advanced" mode. In this mode, event type decorations specified the All satellites column are applied by default, unless satellite-specific decorations are set in the per-satellite columns of the hierarchical view.

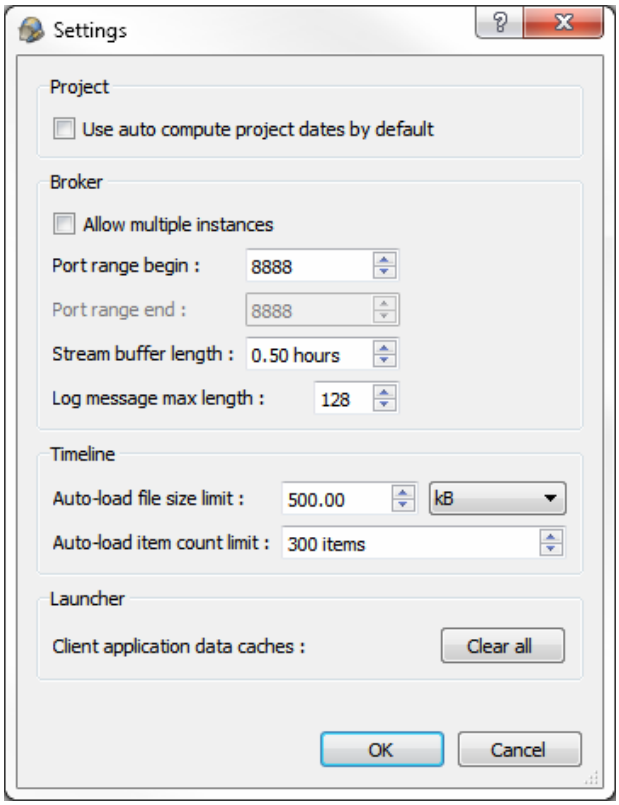


Entity-based event type decorations

Note: All custom satellite-specific decorations will be lost when switching from "advanced" to "basic" mode. When loading a VTS project with satellite-specific decorations, the "advanced" mode is automatically selected.

6.1.6. Settings dialog

General options independent from the project can be set in the File/Settings... menu item.



Settings menu

6.1.6.1. Project options

6.1.6.1.1. Use auto compute dates by default

By default this option is unchecked : the compute dates button must be pressed to take in consideration all data files in use for determining the start and end dates of the project . Check this option to automatically compute date on each visualization run, but it might take a while on large data files.

6.1.6.2. Broker options

6.1.6.2.1. Allow multiple instances

By default, the broker server runs on port 8888. If a visualization is run on a second instance of VTS on the same computer, the Broker will fail. Check this option to fill a range of ports and when the Broker starts, it will scan the range for an open port.

Let this option unchecked but change the port range begin value to force a single value different from 8888.

Important : If the server port is different from 8888, the launchers will be started with a --serverport <portNum> option in command line.

6.1.6.2.2. Stream buffer length

Configure the length of the DATA buffer, which is used to display streamed values in the Timeline.

6.1.6.2.3. Log message max length

Configure the message max width displayed in the Server/Received packets or Server/Sent packets. Wider messages are truncated.

6.1.6.3. Timeline

6.1.6.3.1. Auto-load file size limit

Configure the maximum size for files displayed in the project timeline. The data for files above this size will not be loaded: the file will appear as a gray box without tooltips. Note that the dates and file type are still loaded.

Tuning this setting allows to speed up the loading time of the timeline for projects with large files.

6.1.6.3.2. Auto-load item count limit

Configure the maximum number of data lines for files displayed in any mode other than Block mode in the project timeline. Files containing more lines will have their data loaded but displayed in colored Block mode (with tooltips showing the data).

Tuning this setting allows to improve the responsiveness of the timeline for projects with large files.

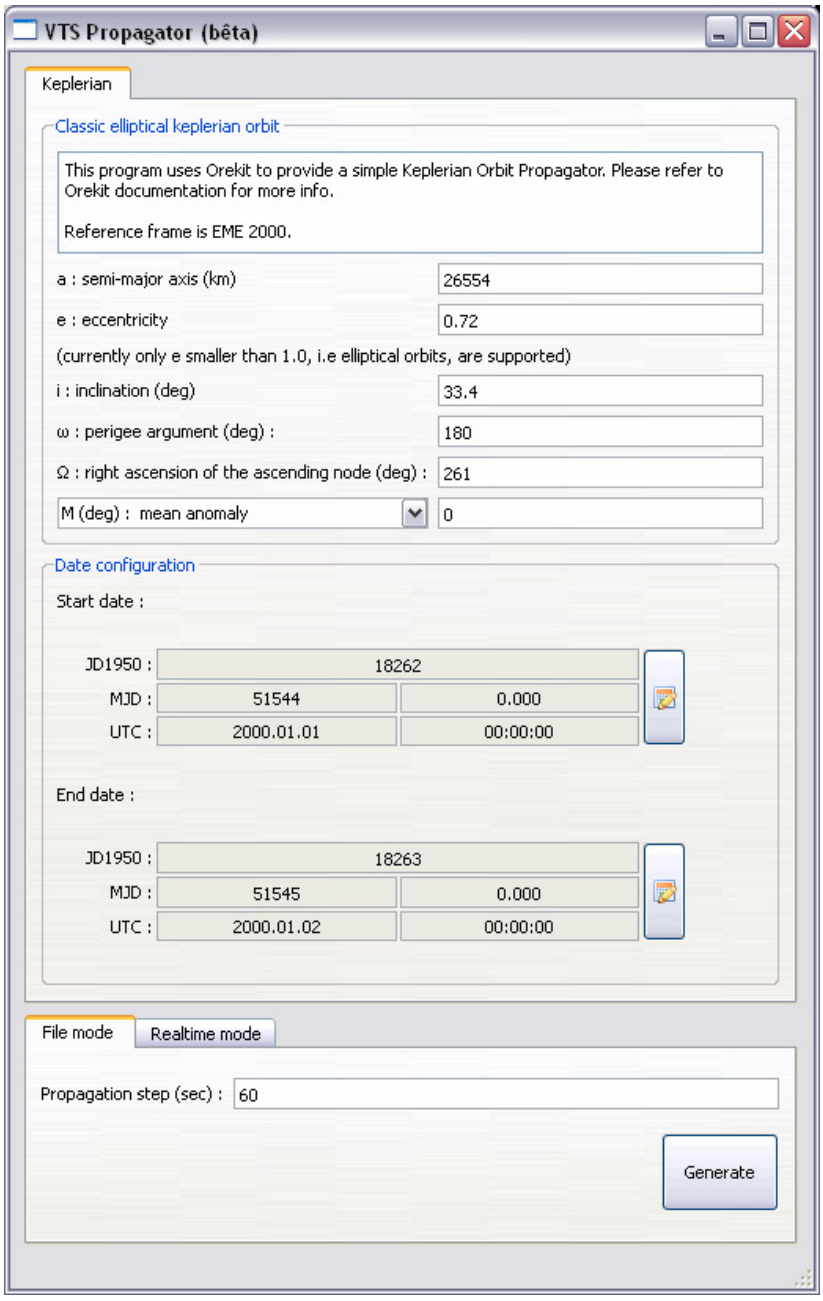
6.1.6.4. Launcher

6.1.6.4.1. Clear all client application data caches

This action triggers an action for all compatible VTS applications that clears the temporary files. This action is available for each client application providing an application cleaner (see documentation in the developer manual)

6.1.7. Orbit propagation with the Propagator

The Propagator can be started with the Start Propagator button in the VTS configuration utility toolbar (1).



VTS Propagator

The Propagator allows automatic generation of CIC/CCSDS files containing position ephemerides for an elliptical keplerian orbit, in a defined time range. Positions are computed in the EME2000 reference frame.

The Propagator relies on the Orekit library to compute the ephemerides. Please refer to this library's documentation for further information on how the computations are carried out.

6.1.7.1. Using the Propagator

6.1.7.1.1. Generating a position file

To generate an orbit ephemerides file, all orbital parameters in the Classic elliptical keplerian orbit area must be filled in. Note that only elliptical orbits are currently supported, i.e. orbits with eccentricity strictly below 1.

The start and end dates for the CIC/CCSDS file must also be defined in the Date configuration area, as well as the time interval between two position values (in seconds).

Clicking the Generate button prompts the user for the output file location and name, then generates the orbit data. The output data file can then be used to define the position of a satellite in the VTS configuration utility.

6.1.7.1.2. Streaming position data

The Propagator is also able to stream position data to the Broker during visualization. Please refer to the Real-time VTS section in the Synchronization protocol for VTS clients chapter for further information on data streaming in VTS.

Streaming mode in the Propagator

In this mode, the user must define a stream ID (e.g. pos) and a time ratio. The Propagator starts streaming the generated position data and time data at a 1Hz rate when the Broadcast button is clicked.

6.2. BROKER USER MANUAL

The Broker is the core application of the visualization phase. It starts the client applications, regulates the visualization time, sends user commands to the client applications, and much more.

6.2.1. Command-line arguments

The Broker is started by the VTS configuration utility when the Run button is clicked. It can also be started from the command line, either directly or through the startVTS binary.

When started from the command line, the following arguments can be specified:

6.2.1.1. --project <File.vts>

This argument tells the Broker which VTS project file to load and visualize.

Example:

```
broker --project C:\Project\CubeSat.vts
```

6.2.1.2. --specificargs <appIdName> <SpecificArgs>

This argument defines additional arguments which will be passed on to client applications upon startup.

Client applications can be identified in several ways:

- The application ID (1, 2, etc.)
- The application name ("Celestia", "2dWin", etc.)

- If no number is specified, then the given arguments will be passed on to all clients with the given name (i.e. all instances of Celestia, 2dWin, etc.)
- If an additional number is specified ("Celestia:1", "PrestoPlot:3", etc.), then the given arguments will only be passed on to the n-th instance of the application

Examples:

```
broker --project C:\Project\CubeSat.vts --specificArgs 1 "--someOption"
broker --project C:\Project\CubeSat.vts --specificArgs Celestia "--url exampleUrl"
broker --project C:\Project\CubeSat.vts --specificArgs 2dWin:2 "--colorLayer"
```

The *--specificArgs* arguments must be specified after the *--project* argument on the command line.

6.2.2. Start of a visualization

When a visualization is started from the VTS configuration utility, the Broker starts all the client applications defined in the VTS project. During this initialization phase, the Broker displays the Initializing message in the text fields of the time control area. Time does not start flowing until all client applications have signaled they are ready.

Upon connection of the various client applications, the Broker's tabs are populated with commands and information regarding these applications.

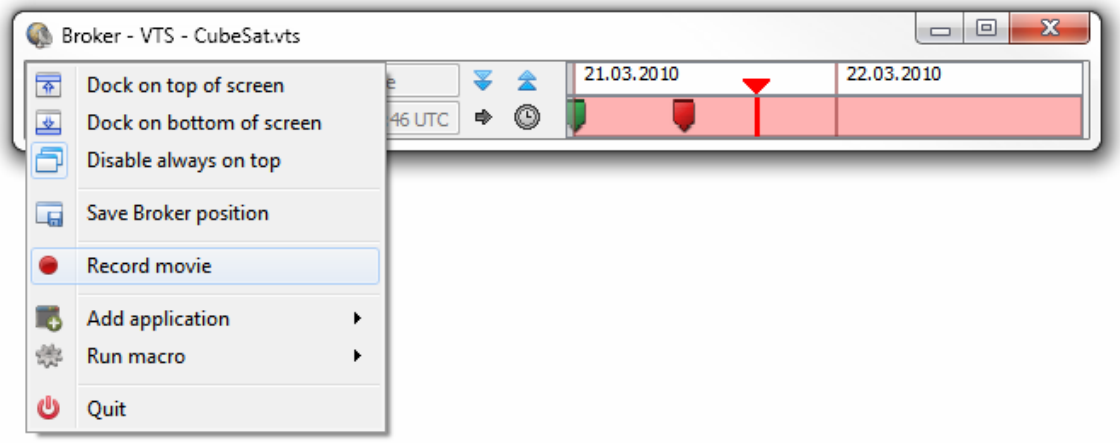
Visualization automatically starts to play with time ratio 1 once initialization is finished, unless specified otherwise in the VTS project.

6.2.3. End of a visualization

The visualization can be stopped directly by clicking the top-right cross button to close the Broker. It also stops once all client applications have been closed by the user, or if the VTS configuration utility is closed (if the visualization was started from there).

6.2.4. Broker menu

The Broker menu can be reached through the gears icon in the top-left corner of the Broker.



Broker menu

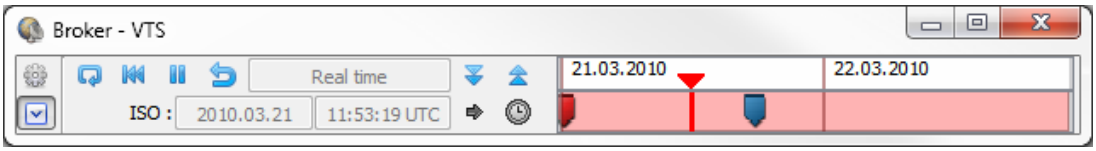
It offers the following options:

- **Dock on top of screen** / **Dock on bottom of screen** / **Enable always on top**: Control special Broker display modes (refer to the Display modes section below for more information)

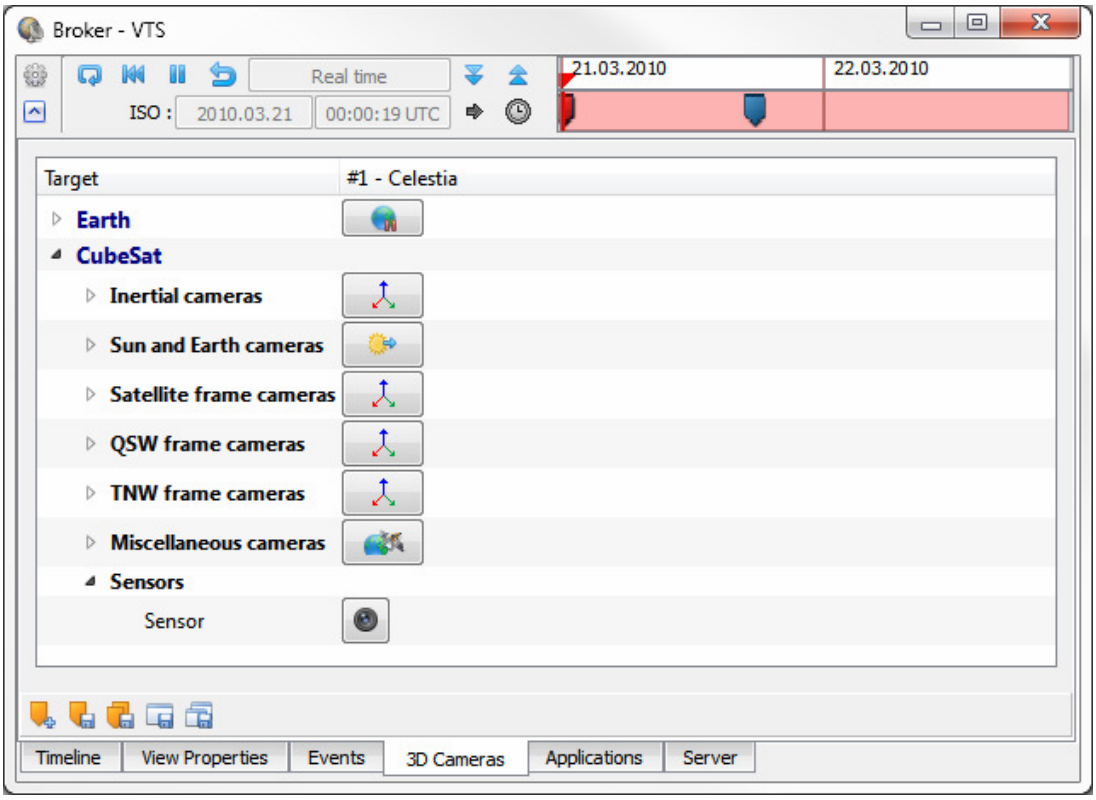
- **Save Broker position**: Save the Broker window state and position, so that it can be restored at the next startup of project visualization
- **Record movie**: Record a movie (refer to the Recording movies section below for more information)
- **Add application**: Start a new dynamic client application (refer to the *Applications* tab section below for more information)
- **Run macro**: Run a CIC/CCSDS macro file (refer to the Scripts and macros in VTS chapter for more information)
    - **Choose macro...**: Browse for a CIC/CCSDS macro file to run
    - **Project macros**: Run a CIC/CCSDS macro file found in the *Macros* subfolder of the project folder
    - **VTS macros**: Run a CIC/CCSDS macro found in the *Apps/Broker/macros* subfolder of the VTS installation folder
- **Quit**: Close the visualization

6.2.5. Display modes

The Broker can either display itself in compact or full mode. In compact mode, only time control buttons are displayed and the Broker is pinned in the foreground (always on top of other windows). In full mode, the tabbed interface is displayed, allowing access to the scenario timeline, all client commands, and information on the state of the visualization. By default, at the start of a visualization, the Broker is in compact mode. The Unfold arrow button allows switching to full mode.



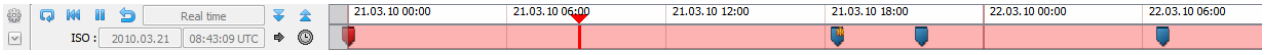
Compact display mode



Full display mode

The gears menu in the top-left corner of the Broker offers the following options:

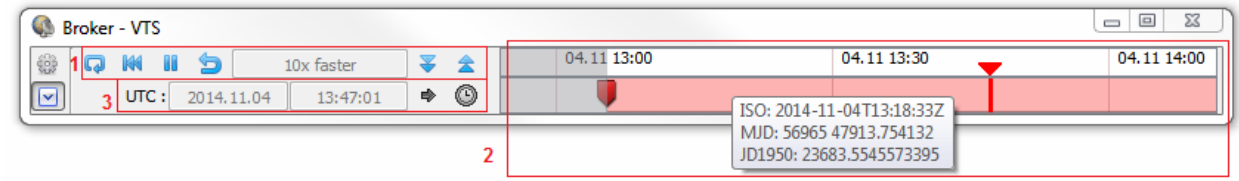
- **Dock on top of screen** : dock the Broker in compact mode at the top of the current screen
- **Dock on bottom of screen** : dock the Broker in compact mode at the bottom of the current screen
- **Enable always on top** : whether or not the Broker window should remain always on top of other windows (enabled by default in compact mode)



Docked display mode

6.2.6. Time management

Controls and information regarding time are available in all Broker display modes. The following interface allows interacting with visualization time:



Visualization time interaction areas

6.2.6.1. Time controls

The buttons in area (1) control time in all client applications:

- **Repeat**: Toggles loop playback for the visualization.
- **Restart**: Resets time to the project's start date. The play/pause status remains unchanged.
- **Play**/**Pause**: Plays/pauses the time flow. Pausing does not prevent interaction with the visualized entities in the client applications, or navigation in visualization time using the timeline.
- **Slower**: Decreases the time ratio. Two clicks on this button result in a 5 times slower time flow. The current time ratio is displayed next to this button.
- **Faster**: Increases the time ratio. Two clicks on this button result in a 5 times faster time flow.
- **Revert**: Every click on this button inverts the time flow direction. The time ratio remains unchanged.

6.2.6.2. Timeline

The timeline (2) displays the visualization's time range. Dragging the cursor controls the visualization's current time. While moving the cursor, the current time is kept updated in all client applications synchronously. For finer or coarser-grained time control, the timeline can be zoomed in/out using the mouse wheel.

6.2.6.3. Time information

A text area (3) displays the current visualization time. The default format is the ISO time format (date and time in UTC). The arrow button circles through other available time formats: CNES julian day (JD1950, fractional days, with reference date January 1st, 1950) and modified julian day (MJD, days and seconds, reference date November 17th, 1858). The Edit date... button pops up a dialog in which the current visualization time can be accurately defined (in all of the above time formats).

Refer to the Date formats in VTS for further information on dates.

6.2.7. Interacting with client applications

The Timeline tab displays graphical information on the progress of the visualization.

The View Properties and 3D Cameras tabs allow interacting with client applications. Available actions are hierarchised according to the project structure, for each application.

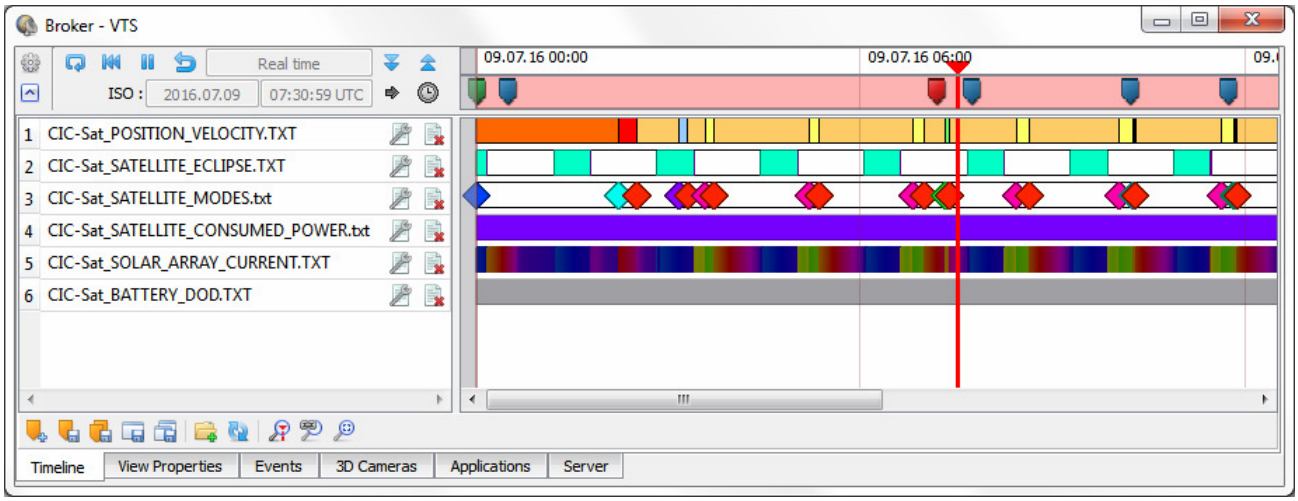
The Events tab allows controlling the visibility of events in client applications. Event types are displayed in a tree structure for applications that support events.

The Applications tab allows managing the Broker's client applications.

The Server tab displays information, warning and error messages from the Broker or its client applications, provides a log of all messages received by the Broker and sent to its clients, and displays some technical information on all currently connected clients.

6.2.8. Timeline tab

The Timeline tab displays graphical information on the progress of the visualization.

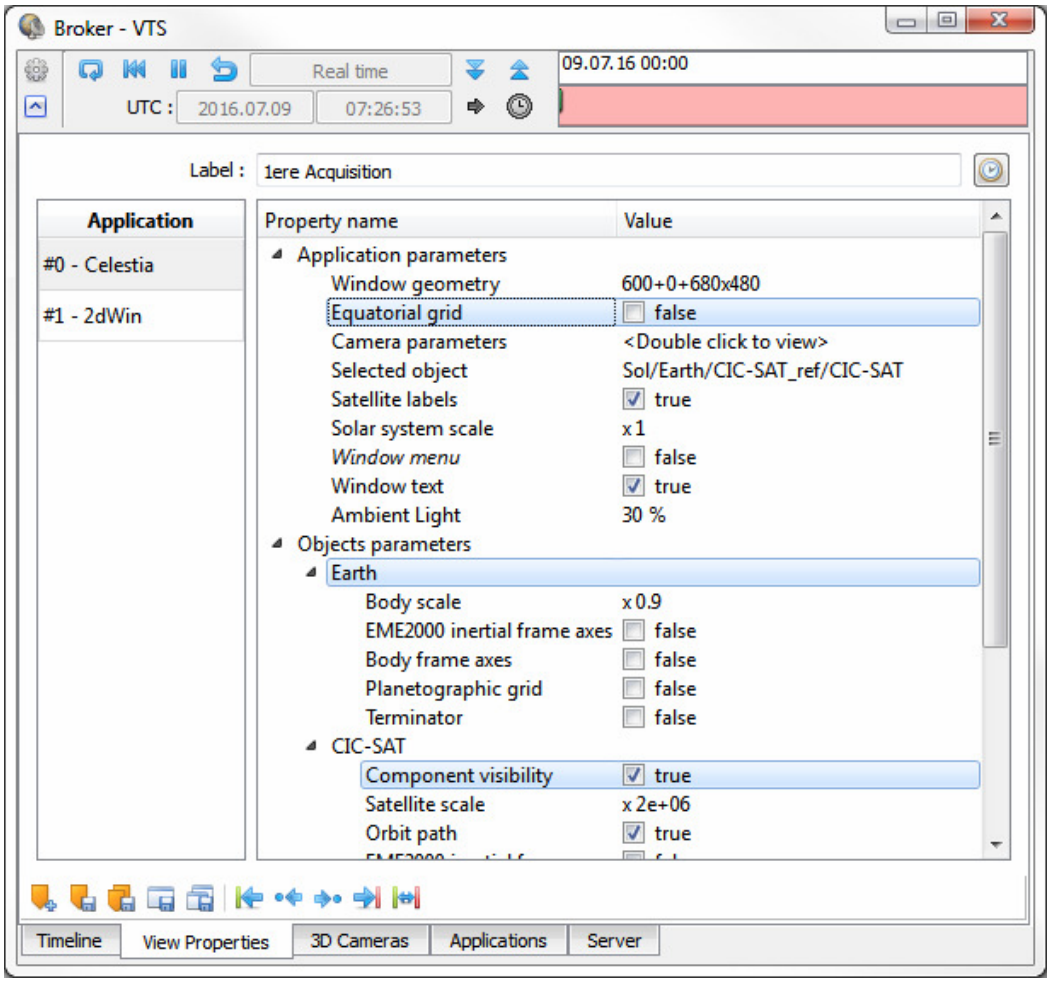


Project timeline

The contents of the timeline are described in the Timeline section of the Scenario in VTS chapter.

6.2.9. View Properties tab

The View Properties tab allows interacting with client applications and defining the properties of all project scenario states.

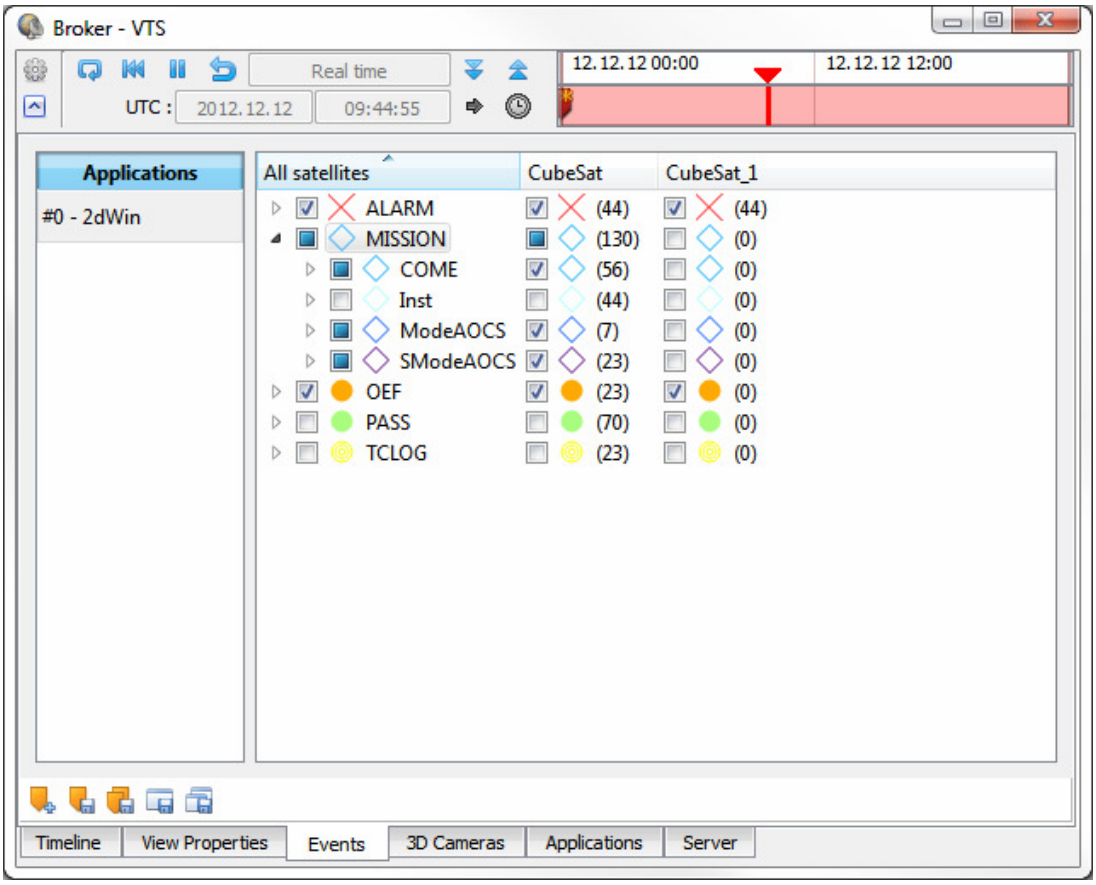


Client application view properties

The contents of this tab are described in the View properties editor section of the Scenario in VTS chapter.

6.2.10. Events tab

The Events tab allows controlling the visibility of mission events in client applications that support them.



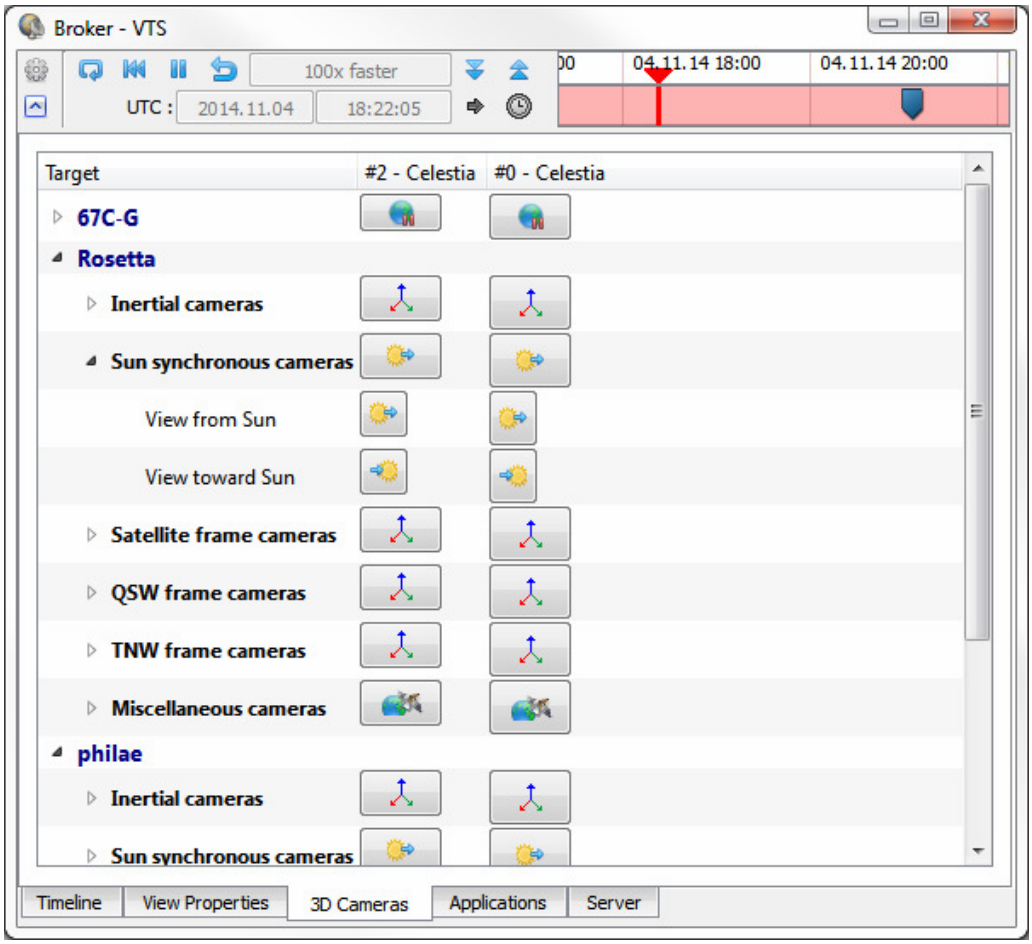
Mission events visibility

- The list of client applications allows selecting the target client when setting the visibility of mission events.
- The event type hierarchy displays all event types and allows setting the visibility of an event type for all satellites or for each satellite individually. The value indicated in parentheses is the number of events of the corresponding event type in all event files.
- The visibility status of all event types is maintained in the scenario states, and is hence saved between states and visualization sessions.
- Mission events are also displayed in the *Timeline* tab, for each satellite.

For more information on events in VTS, refer to the Mission events in VTS chapter.

6.2.11. 3D Cameras tab

The 3D Cameras tab allows interacting with standard visualization cameras in 3D applications. The various cameras are displayed in a hierarchical fashion for all visualization entities, and each column corresponds to a specific instance of a currently running 3D client application.



Camera controls

6.2.11.1. Central bodies

Central bodies such as Earth are top-level entities of the visualized project. All central bodies defined in the project appear in the hierarchy. Available actions are:

- **Fixed in *Body* frame**: The camera is positioned in the body's local frame, pointed at the body.
- **Inertial**: The camera is positioned in an inertial frame attached to the body, pointed at the body.
- **Goto**: The camera is pointed at the body, travelled so that it occupies as much space as possible in the 3D window, and attached to the body's local frame. The UP vector of the camera is undefined.
- **Center**: The camera is simply pointed at the body. Its reference frame is unchanged. The UP vector of the camera is undefined.
- **North pole**: The camera is positioned above the North pole of the body, pointed towards it, and attached to the body's local frame. The UP vector of the camera is along the Y axis of the local frame.
- **South pole**: The camera is positioned above the South pole of the body, pointed towards it, and attached to the body's local frame. The UP vector of the camera is along the -Y axis of the local frame.

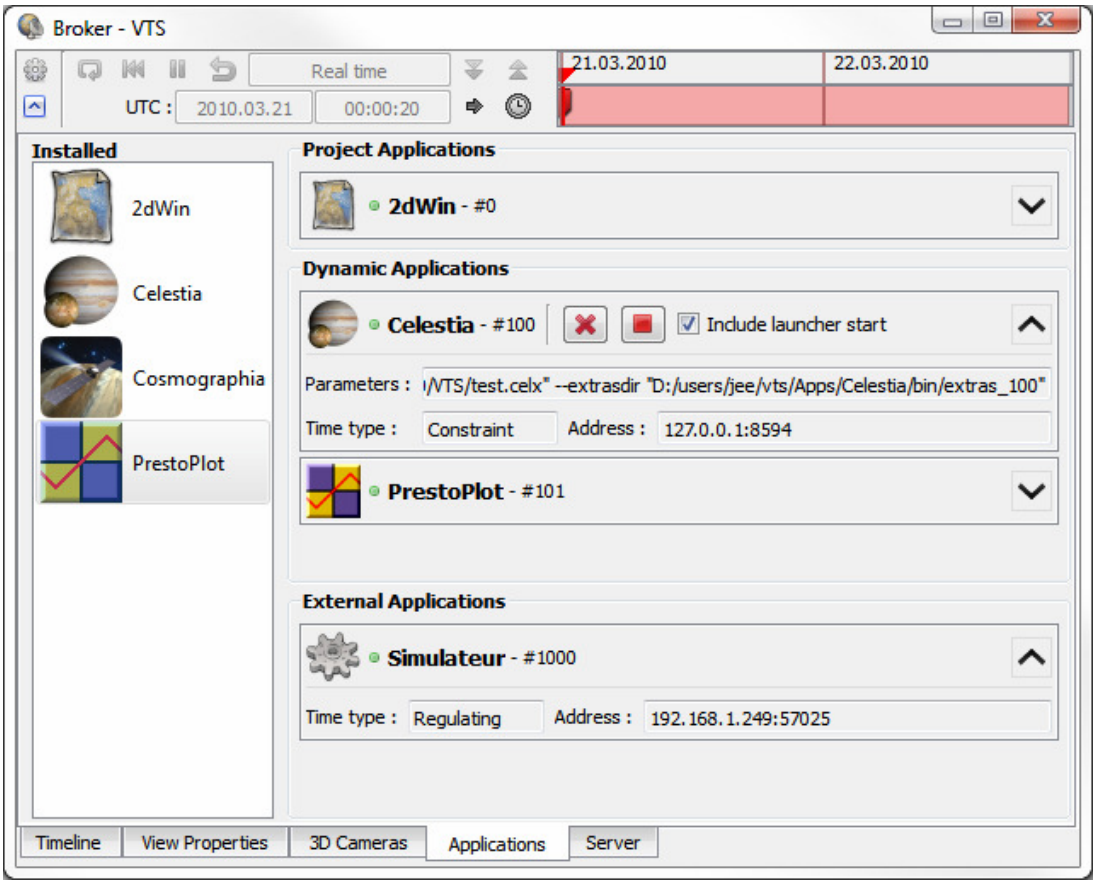
6.2.11.2. Satellites

Satellites are top-level entities of the visualized project. All satellites defined in the project appear in the hierarchy. Available actions are:

- **Inertial cameras**: The camera is positioned in an inertial frame attached to the satellite, pointed at the satellite. Expanding this item allows positioning the camera on all axes of the inertial frame.
- **Sun and *Body* cameras**: The camera is positioned in a Sun/body-synchronous frame attached to the satellite, pointed at the satellite. Expanding this item allows positioning the camera so that it points towards the Sun/body. The UP vector of the camera is undefined.
- **Satellite frame cameras**: The camera is positioned in the satellite's local frame, pointed at the satellite. Expanding this item allows positioning the camera on all axes of the satellite frame.
- **QSW frame cameras**: The camera is positioned in the QSW local orbital frame, pointed at the satellite. Expanding this action allows positioning the camera on all axes of the QSW frame.
- **TNW frame cameras**: The camera is positioned in the TNW local orbital frame, pointed at the satellite. Expanding this action allows positioning the camera on all axes of the TNW frame.
- **Miscellaneous cameras**
    - **Orbit**: The camera is positioned at a distance of the satellite's central body, pointed towards the body in a direction normal to the satellite's velocity, so that the orbit of the satellite can be observed. The UP vector of the camera is aligned on the Z axis of the central body's local frame. Hold the CTRL key while clicking to observe the scene from the opposite side of the body.
    - **Goto**: The camera is pointed at the satellite, travelled so that it occupies as much space as possible in the 3D window, and attached to the satellite's local frame. The UP vector of the camera is undefined.
    - **Center**: The camera is simply pointed at the satellite. Its reference frame remains unchanged. The UP vector of the camera is undefined.
- **Sensors**: These cameras are only available when sensors are attached to the satellite. The camera is positioned at the location of the sensor (plus an offset which can be set in the application parameters), pointed in the direction of its aim vector. Expanding the action allows selecting each sensor of the satellite. The UP vector of the camera is the X axis of the sensor. Hold the CTRL key while clicking to use the Y axis as the UP vector.

6.2.12. Applications tab

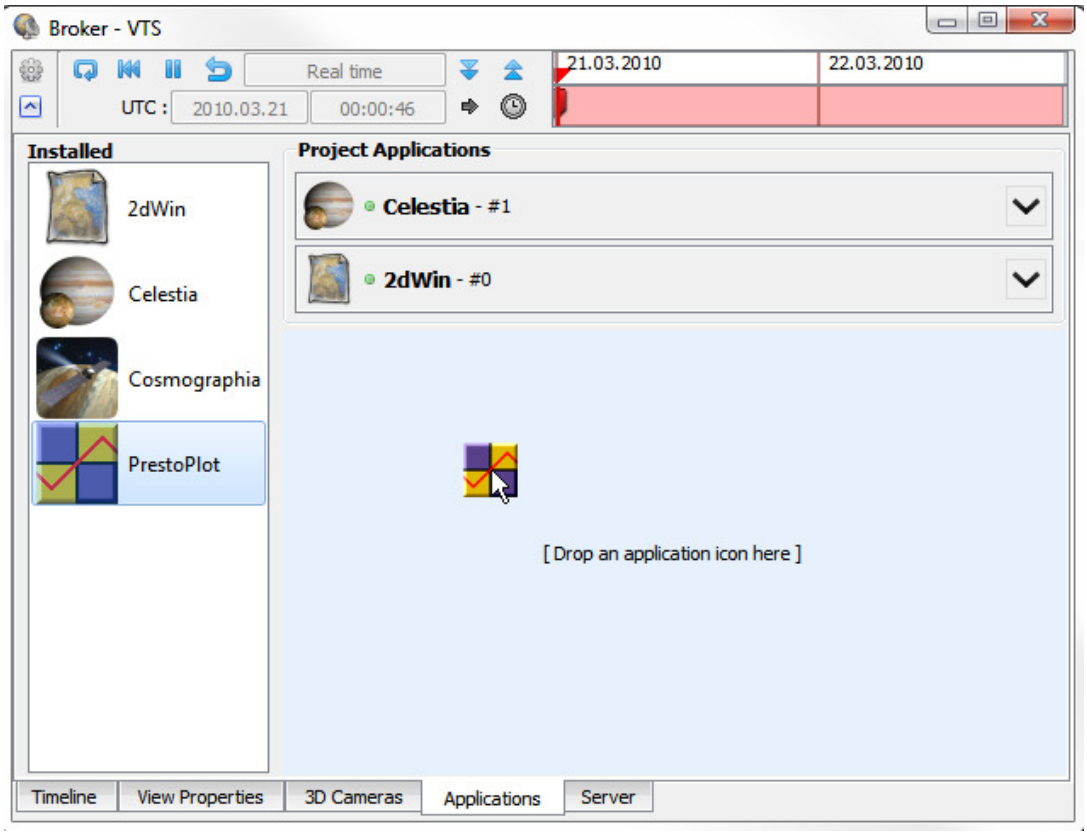
The Applications tab allows managing currently connected client applications, as well as starting new clients.



Client applications

6.2.12.1. Managing client applications

- To start a new instance of a client application, double-click its icon in the *Installed* list or drag-and-drop it into the right area of the window. Clients started this way are listed in the *Dynamic Applications* category.
- Clients not started by the Broker but connected to it are listed in the *External Applications* category.
- More details about a client can be displayed by clicking the arrow button next to each client.
    - To stop a client, click the red **Stop** button.
    - To restart a stopped client, click the green **Play** button. A checkbox allows specifying whether or not to execute the client application's launcher before restarting the application, in order to update the client's data.
    - To remove a dynamic application from the list, click the red cross button. The client will be stopped if it is currently running.



Starting a client by drag-and-drop

6.2.12.2. Promoting a dynamic application into a project application

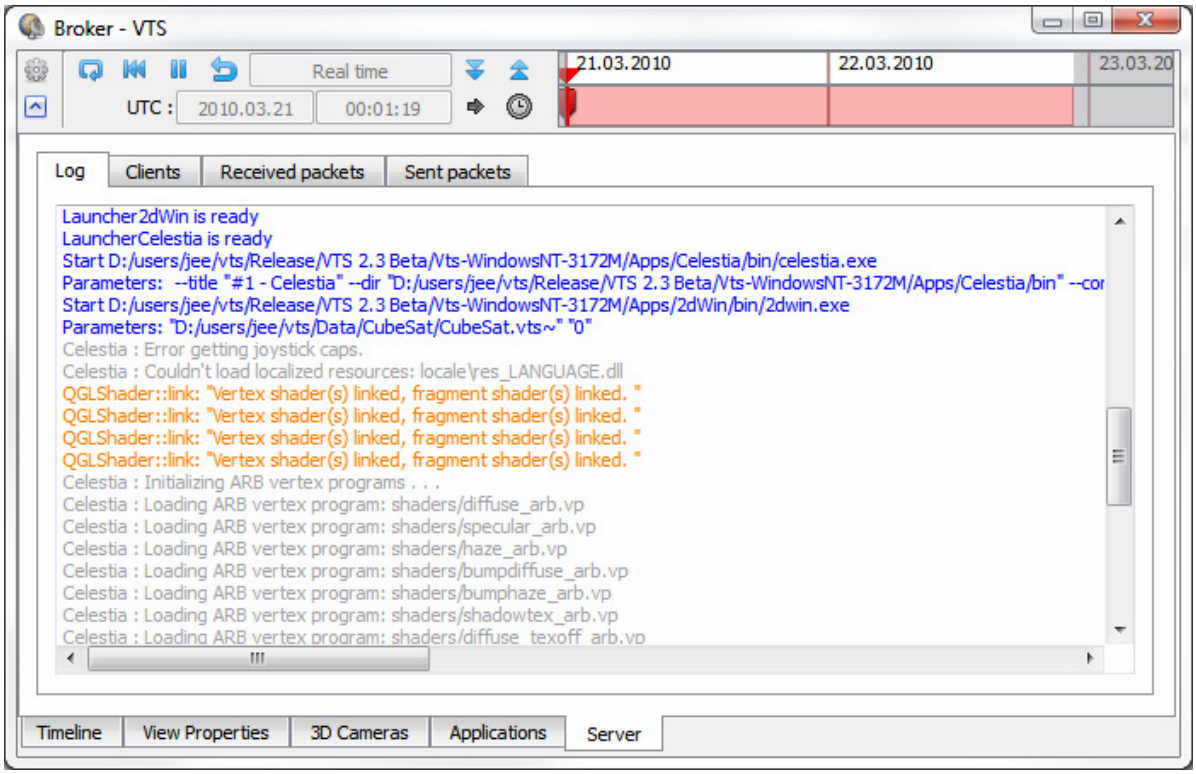
Dynamic applications can be promoted to project applications by drag-and-dropping their icons into the Project Applications list. When closing the Broker, a pop-up dialog will offer to save the changes made to the VTS project. If changes are saved, the promoted client will appear in the list of project applications in the VTS configuration utility and its view properties will be saved in the project scenario.

6.2.13. Server tab

The Server tab displays information on the connection and communication of client applications with the Broker.

6.2.13.1. Log tab

The Log tab displays the log of all messages emitted by the Broker or client applications during visualization. Since messages are collected from applications executed in different processes, the order in which the messages are displayed should not be taken as an accurate representation of the order in which the messages have actually been emitted.



Logged messages

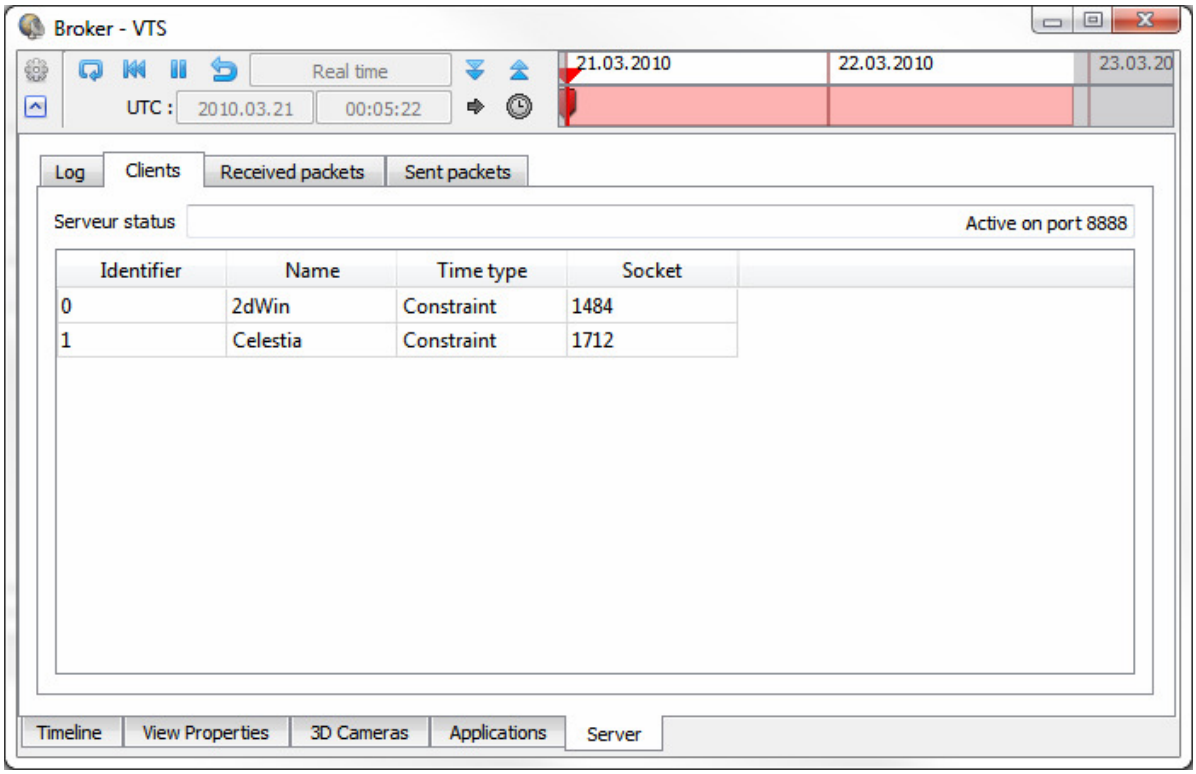
The color code of logged messages is as follows:

- Blue: informational messages about the state of the visualization. Only emitted by the Broker.
- Orange: warning messages. Only emitted by the Broker. These messages usually appear when an action fails, without causing the visualization to stop.
- Red: error messages. Only emitted by the Broker. These messages usually appear when a client application crashes.
- Grey: messages printed to the standard output of client applications. These messages are prefixed with the name of the client application that emitted them. They can be either informational, warning or error messages.

Note: Always remember to send a copy of the message log to the VTS support team when an error occurs during visualization.

6.2.13.2. Clients tab

The Clients tab displays the connection state of client applications.



Connection state of client applications

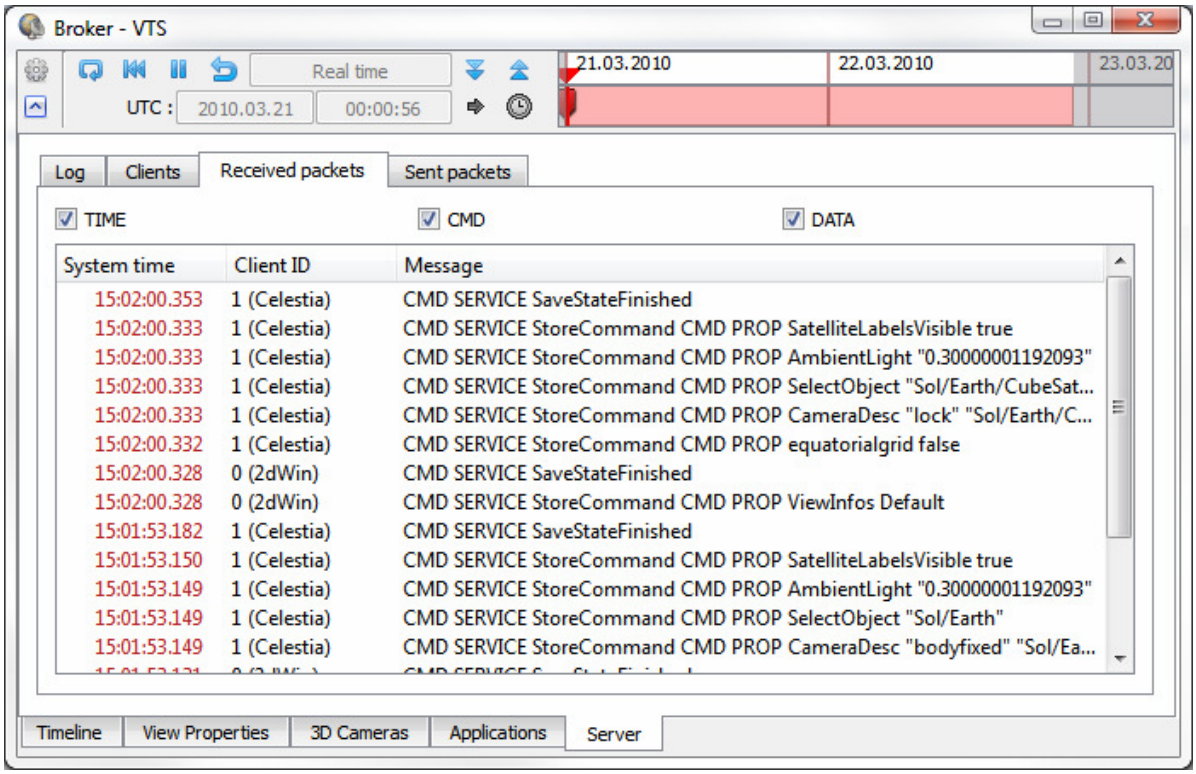
- **Server status**: indicates the TCP port used by the Broker.

The following information is available for each client:

- **Identifier**: the client's ID
- **Name**: the client application's name
- **Time type**: the time behaviour of the client (*Constraint* or *Regulating*). Refer to the Synchronization protocol for VTS clients section for further information.
- **Socket**: the socket ID for Broker-client communication.

6.2.13.3. Received packets and Sent packets tabs

The Received packets and Sent packets tabs display the messages respectively received and sent by the Broker, from or to client applications.



Messages received from clients

TIME messages deal with time synchronization of clients, while CMD messages are related to client commands. All messages are displayed truncated when they are too long. Further information on these messages can be found in the Synchronization protocol for VTS clients section.

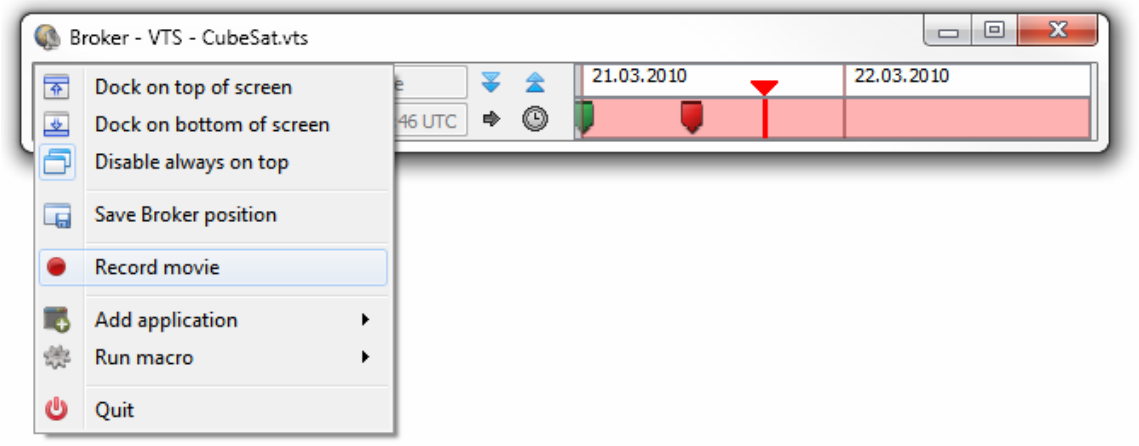
The TIME, CMD and DATA checkboxes allow enabling or disabling the display of these classes of messages in the list.

6.2.14. Recording movies

Movies can be recorded by the Broker during visualization. When recording a movie, client applications are synchronized frame by frame and an image is captured at each frame. The output movie is set at a framerate of 25 images per second.

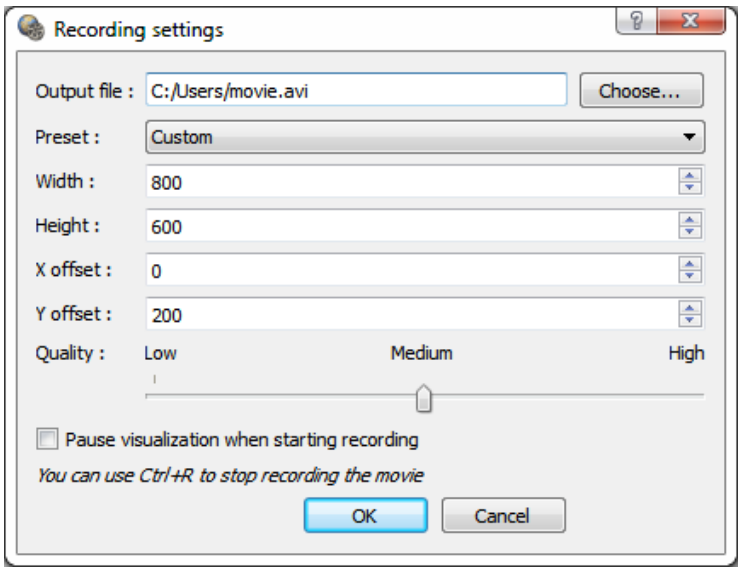
Note that recording a movie is not possible when the Broker is in real-time mode. For more information on real-time mode, refer to the Real-time VTS section of the Synchronization protocol for VTS clients chapter.

To start recording a movie, click the Record movie entry in the Broker's action menu:



Recording a movie

A configuration dialog pops up and visualization is paused. Interaction with the Broker is disabled while the dialog is opened, which means that the visualization must be correctly configured before starting the recording.



Recording settings

The following parameters must be provided:

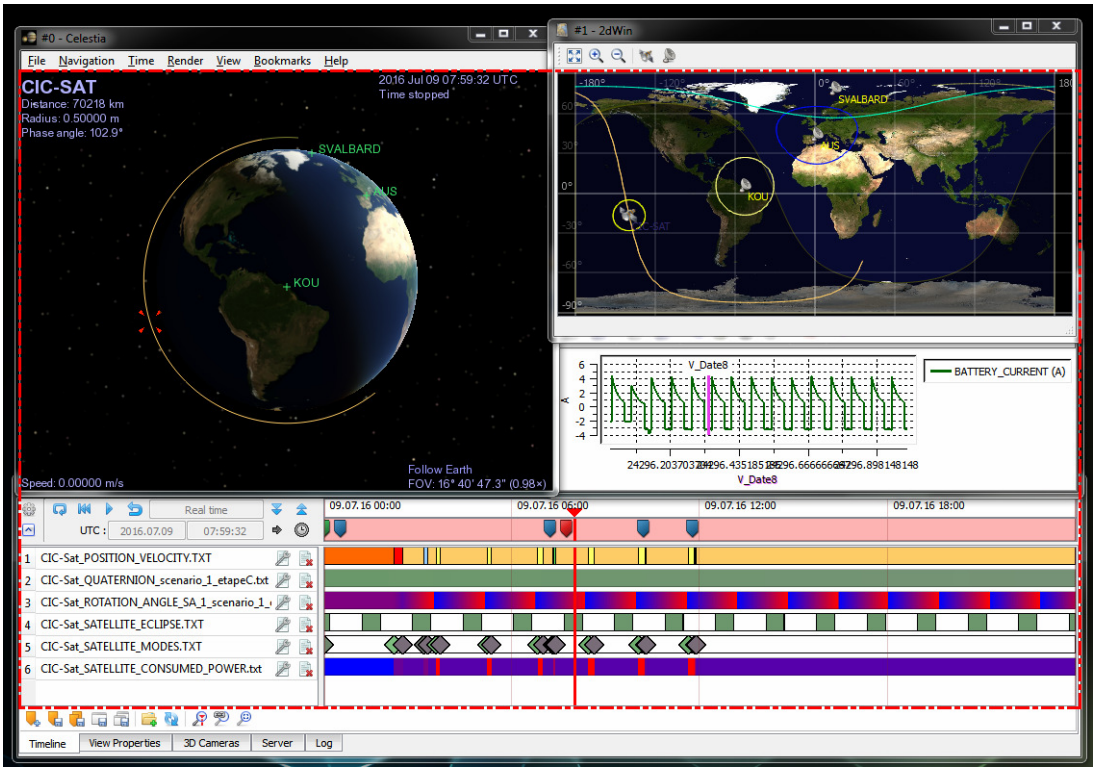
- **Output file**: Name of the movie output file. The file extension will be either *.avi* (MPEG-4) or *.mpg* (MPEG-2). For correct integration of the recorded movie within Microsoft PowerPoint, the *.mpg* format must be selected.
- **Preset**: List of standard movie definitions. A *Custom* definition allows manual specification of the movie width and height.
- **Width**, **Height**: Movie definition (the mouse wheel may be used on these fields for faster adjustment of the values). For technical reasons linked to the requirements of the encoding codec of the movie, these values must be multiples of 8. If not, they will be automatically rounded up to the nearest multiple of 8 at the start of the recording.
- **X offset**, **Y offset**: Position of the recorded area (the mouse wheel may be used on these fields for faster adjustment of the values).

- **Quality** (*Low*, *Medium*, *High*): Quality factor determining the bitrate of the recording.
- **Pause visualization when starting recording**: Option allowing to start the recording paused at the current visualization time.

Notes:

- The movie bitrate is proportional to the movie definition.
- The quality factor increases or decreases the quality of the movie. A *Low* quality factor produces compression artifacts on moving parts in the video, but smaller file size. A *High* quality factor produces good quality video, but significant file size.

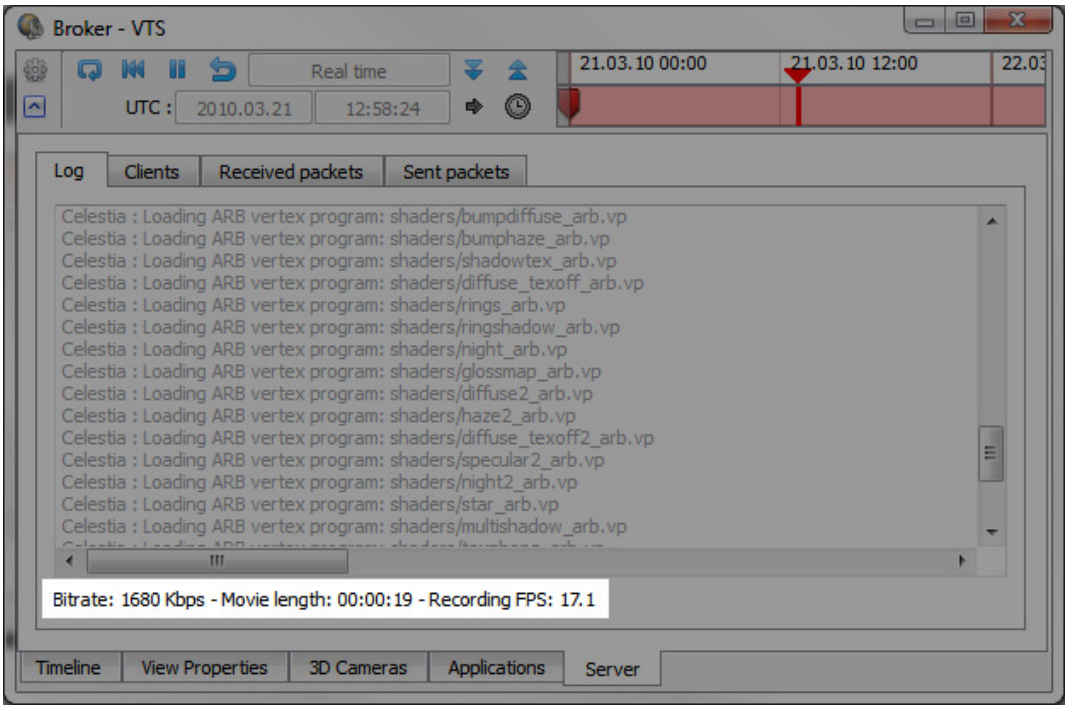
The recording can be stopped with the Stop recording entry in the Broker's action menu. The Ctrl+R keyboard shortcut can also be used when the Broker window is focused.



Recording area

The screen area located within the red frame is recorded in the movie. The mouse pointer is not recorded. During the recording, it is advised not to move recorded windows and to use camera states rather than manual camera movements. Otherwise, since the movie is not recorded at real-time speed, camera movements may appear to be slower or faster than expected.

During the recording, additional information is displayed in the Log tab of the Server tab of the Broker:



Recording information

- **Bitrate**: the current movie bitrate
- **Movie length**: the current length of the recorded movie
- **Recording FPS**: the current record framerate (this does not affect the output framerate)

6.2.15. Other interactions

The Ctrl+Shift+C keyboard shortcut copies to the clipboard the full path to the VTS project file.

6.3. 2DWIN USER MANUAL

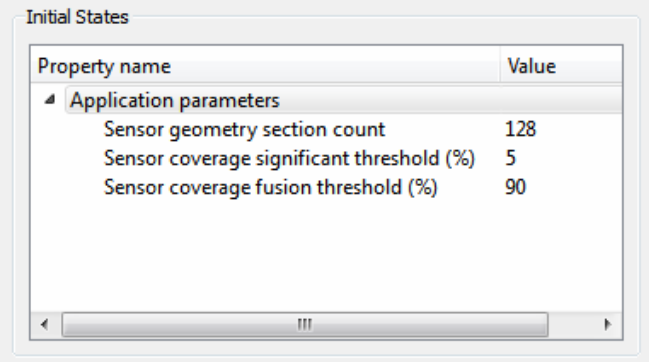
The 2DWin client application displays a planisphere of the project's main central body in plate carrée projection.

6.3.1. Configuration in the VTS configuration utility

The main central body displayed in 2DWin is the first central body defined in the project hierarchy in the VTS configuration utility. Refer to the Configuring a central body section in the VTS configuration utility user manual chapter for more information. Only satellites attached to this main body will appear in the 2D view.

6.3.2. Specific application parameters in VTS

When adding 2dWin as a VTS client application, some parameters can be set by clicking on the 2dWin entry in the VTS project tree:



Celestia application parameters

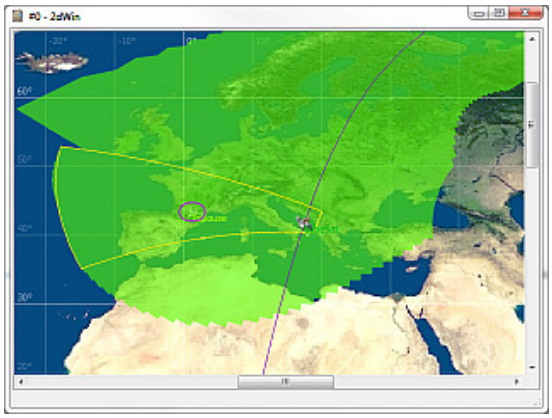
- **Sensor geometry section count**: Number of points making up the outline polygon of the aiming sensor surface. Performance and display accuracy depend on this parameter.

One the one hand, performance may be affected by a too high setting and a long residual trace. On the other hand, accuracy is better with a high setting when the satellite attitude has a large angle with the nadir.

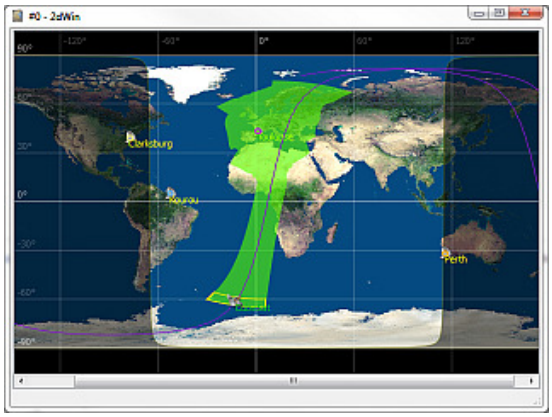
If you want to focus on sensor swath accuracy: increase the SensorGeometrySectionCount parameter and favor a short residual trace. If you want to view long missions coverage: lower the SensorGeometrySectionCount parameter with a long residual trace.

- **Sensor coverage significant threshold**: The sensor swath is composed by a set of instantaneous sensor aiming surfaces represented by polygons. An instantaneous aiming surface is retained when a significant percentage of new coverage is provided by the new surface. The SensorCoverageSignificantThreshold parameter sets the percentage of novelty. This is a minimum threshold.

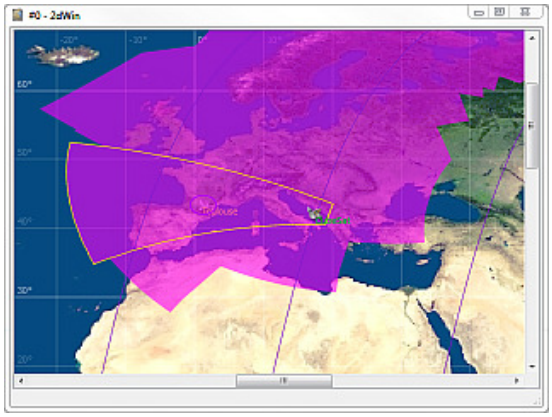
Use a small value for an agile satellite. Use a high value for a long coverage mission.



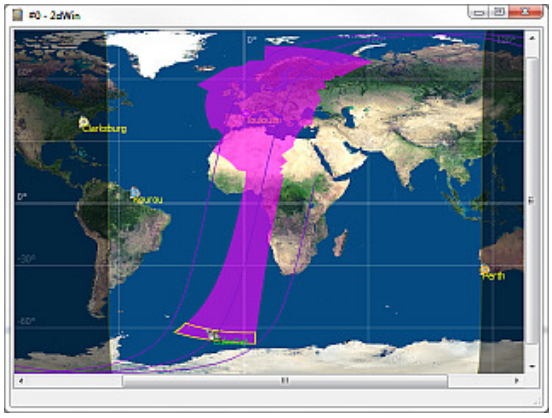
A significant threshold of 5% of novelty is ideal for an agile satellite maneuver



A small threshold works well for long coverage missions but slows down the display



A significant threshold of 40% gives poor results during agile satellite maneuvers



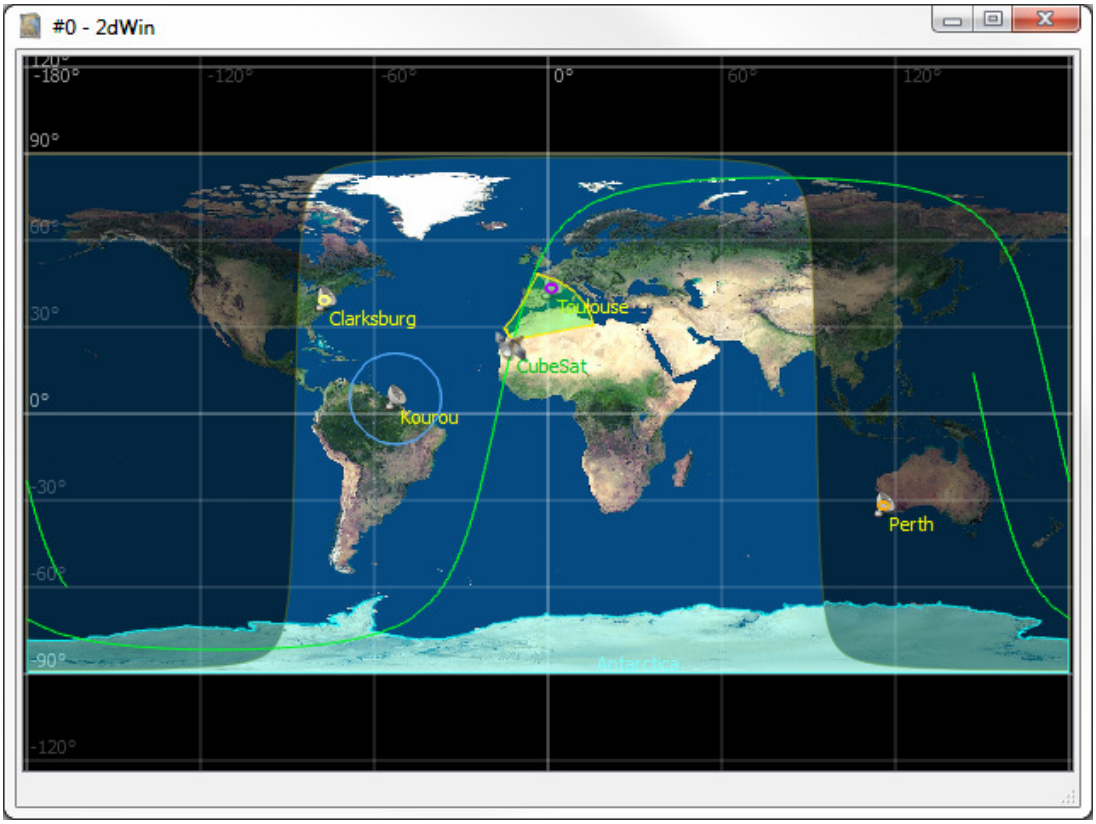
A high threshold gives good results for long coverage missions with good performance

- **Sensor coverage fusion threshold**: The sensor residual trace is composed by many polygons. Its length is maintained by deleting polygons at the end of the tail. For better performance, polygons are merged together when two polygons have at least a certain percentage of area in common. The SensorCoverageFusionThreshold parameter sets the percentage of novelty. This is a minimum threshold.

Choose a small value when the length of the sensor residual trace must be accurate (0% of fusion will keep all

polygons separated). Choose a high value for better performance (99% of fusion will merge huge blocks of polygons, which will be removed one by one, making the residual trace length vary a lot). Choose the special value of 100% to only merge by orbit path color : this mode is useful for long coverage missions, where only the current orbit pass should be displayed.

6.3.3. Description of graphic items



2DWin

The following items can be seen on the planisphere of 2DWin:

- The **latitude/longitude grid** automatically adapts to the current zoom level. The latitude and longitude values are displayed respectively at the left and top of the window.
- The **terminator** is the fictional line that separates the sunlit and obscured faces of the body. It is displayed in yellow, the obscured face appearing darker than the sunlit one.
- The **subsolar point** is the point at the surface of the body where the Sun is at the zenith.
- The **satellite ground track** appears before and after the satellite. Its length can be defined in the VTS configuration utility. In 2DWin, the track is displayed in the body's local frame (as opposed to an inertial frame in Celestia).
- The **satellite icon** is the projection of the position of the satellite in orbit on the surface of the body.
- The **sensor footprint** is the intersection of the sensor's aim volume with the surface of the body. Refer to the Configuring a satellite sensor section in the VTS configuration utility user manual chapter for more information.

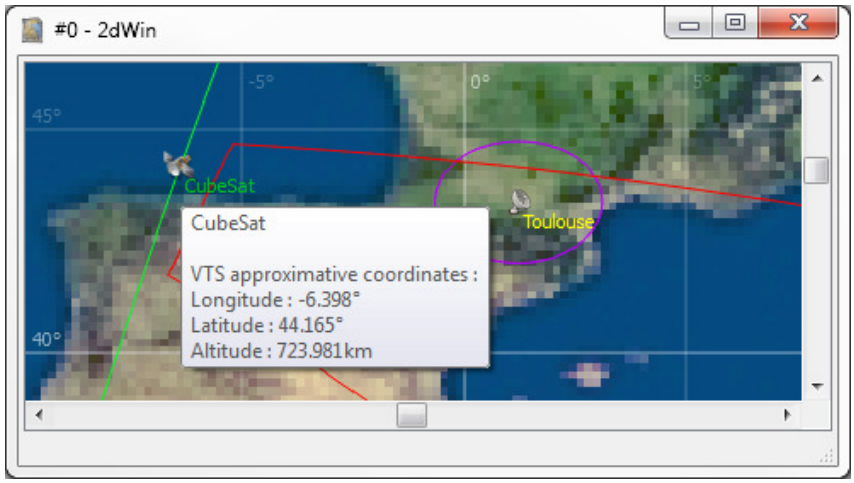
- The **ground stations** attached to the body. Refer to the Configuring a ground station section in the VTS configuration utility user manual chapter for more information.
- The **Regions Of Interest** attached to the body. Refer to the Configuring a Region Of Interest section in the VTS configuration utility user manual chapter for more information.
- The **mission events** attached to all satellites of the body. Mission events appear at the date of their occurrence on the ground track of the satellite they are attached to. Refer to the Events attached to a satellite section in the VTS configuration utility user manual chapter for more information.

6.3.4. Interacting with the application

6.3.4.1. Context menu

The context menu accessible by right-clicking anywhere in the application window offers the entries:

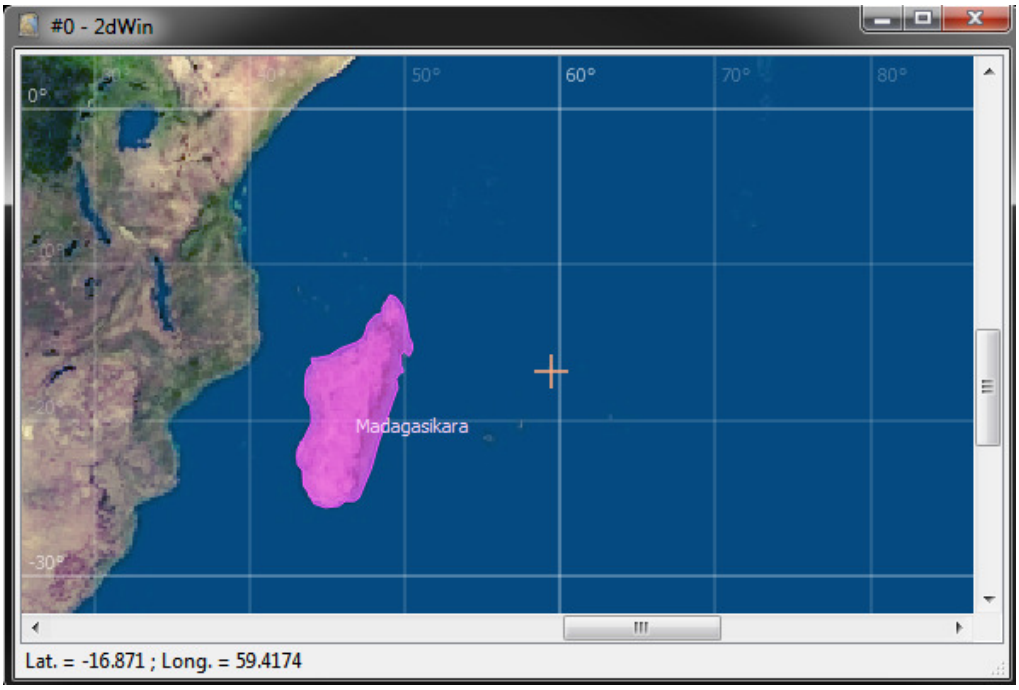
- The **Zoom Fit** entry adjusts the zoom level so that the whole planisphere fits in the window.
- The **Zoom In** and **Zoom Out** entries respectively zoom in an out on the planisphere. This can also be achieved using the mouse wheel. The zoom is then centered on the current mouse position in the window.
- The **Toggle Satellite Visibility** entry shows/hides all satellite icons.
- The **Toggle Ground Station Visibility** entry shows/hides all ground station icons.
- The **Clear All Sensor Swath** entry clears the sensor swath overlay, if enabled in the project.



Zoom on a satellite

6.3.4.2. Cursor coordinates

- The cursor coordinates can be displayed by pressing the **Ctrl key**. The coordinates expressed in latitude longitude are displayed in the window status bar.
- Use **Ctrl + left click** to grab the cursor coordinates into the clipboard. You can grab multiple coordinates by maintaining the Ctrl key pressed and left clicking on different locations. They can be directly pasted into a POI or ROI file.



Grabbing cursor coordinates

6.3.4.3. Interacting with a satellite

6.3.4.3.1. Satellite lock

Clicking on a satellite centers and locks the view on this satellite during the animation. Zoom actions are also centered on the satellite instead of the mouse cursor. Clicking the map unlocks the view.

6.3.4.3.2. Geographical coordinates

Hovering the mouse over a satellite or ground station displays a tooltip containing their geographical coordinates.

These coordinates are computed by VTS and are approximate, especially when computing the longitude near the poles.

6.3.4.3.3. Moving in time

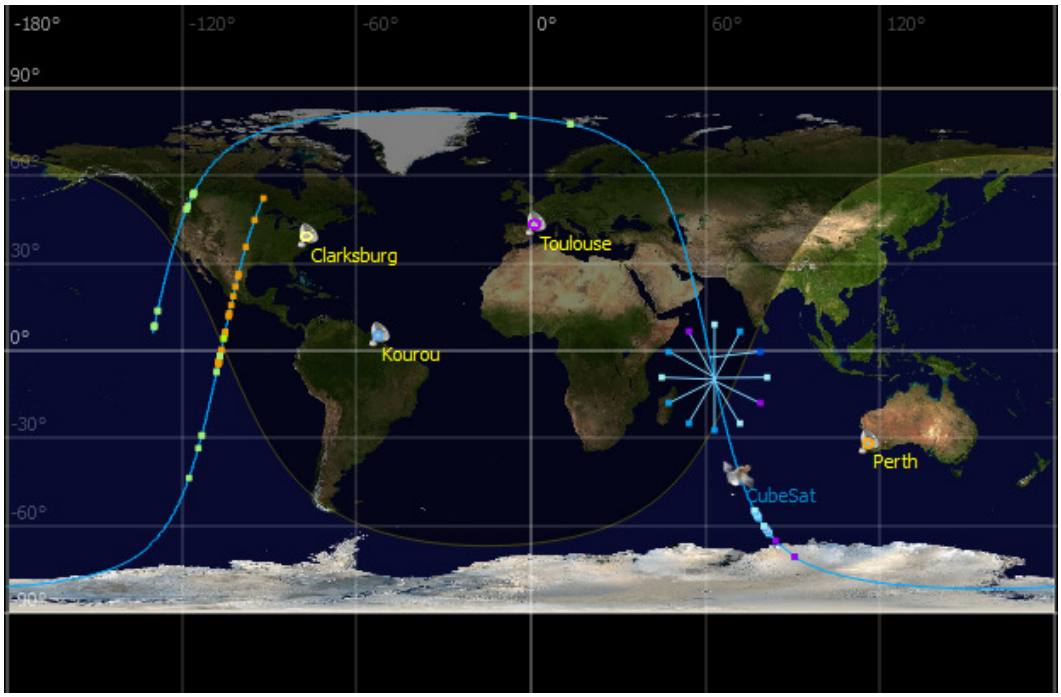
A satellite can be dragged on its ground track to move forward or backward in time. The dragged satellite automatically snaps to the nearest point of its ground track. The corresponding date is sent to the Broker and all applications are synchronized.

It is possible to "jump" from orbit to orbit if the ground track is long enough (greater than the period of the satellite's orbit). This results in a similar "jump" in visualization time.

Notes:

- Visualization time is automatically paused when dragging a satellite.
- Near intersections of ground tracks (typically near the poles, or for ground tracks greater than the period of the satellite's orbit), dragging a satellite becomes very "unstable": the satellite may "jump" from orbit to orbit depending on the position and movement of the mouse. It is advised to avoid dragging a satellite in these areas if "jumping" from orbit to orbit is not desired.

6.3.4.4. Mission events



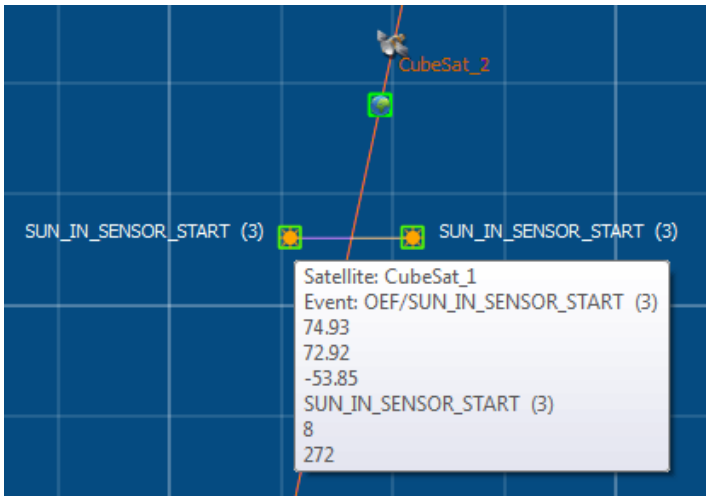
Mission events attached to a satellite

The 2DWin application displays mission events attached to satellites and allows some interaction with them. Refer to the Events attached to a satellite section in the VTS configuration utility user manual chapter for more information.

The appearance of mission events can be defined in the Event Type Editor tab of the VTS configuration utility. Refer to the Configuring event types section in the VTS configuration utility user manual chapter for more information.

6.3.4.5. Interacting with events

- Events at the exact same date -- or at dates close enough that they appear to be at the same date at the current 2DWin zoom level -- will animate and split when the mouse is hovered above them.
- Hovering above a single event will display a tooltip containing the name of the satellite the event is attached to, the name of the event, as well as additional data provided by the CIC/CCSDS event file.
- The line segment linking the icon of an event to its actual position on the satellite ground track is the same color (only a bit lighter) as that of the satellite ground track.
- Double-clicking an event will set the current visualization time to the date of the event. The satellite then appears located at the position of the event. The visualization is also paused.
- The visibility of all event types can be controlled from the *Events* tab of the Broker. Refer to the *Events* tab section of the Broker user manual chapter for more information.



Simultaneous events for two separate satellites

6.3.5. Technical notes

6.3.5.1. Rotation model

2DWin uses the VTS ephemeris catalog to display an accurate positioning and Sun terminator (see the [Central bodies in VTS] section for more information about ephemeris origin). For custom positioning or unsupported bodies, suitable ephemerides files will need to be provided to override the catalog ones. Refer to the Position and orientation of a body section of the VTS configuration utility user manual chapter for more information.

6.3.5.2. Central body texture

The current architecture of the 2DWin application enforces a few constraints:

- The *built-in* texture for Earth is embedded in 2DWin. It is based on the Blue Marble photos from NASA (image link, documentation link), using the equidistant cylindrical (plate carrée) projection.
- For other solar system bodies or celestial bodies available in Celestia, the *built-in* texture will be taken from Celestia.
- For bodies not available in Celestia, a black texture will appear in *built-in* mode. A custom texture should then be defined. Refer to the Texture of a body section of the VTS configuration utility user manual chapter for more information.

6.4. CELESTIA USER MANUAL

The main 3D client application in VTS is Celestia. Celestia is open-source software, developed, amongst others, by Chris Laurel.

VTS relies on the latest source version of Celestia (http://www.shatters.net/celestia). It is interfaced with VTS through LUA scripts.

6.4.1. Integration with VTS

Celestia is fully integrated with VTS. All of Celestia's features related to the visualization of satellites within the perimeter of VTS are available directly from the Broker. These features are described in detail in the Messages received by Celestia section of the Synchronization protocol for VTS clients chapter, and in the 3D Cameras tab section of the Broker user manual chapter.

6.4.2. Navigating in Celestia

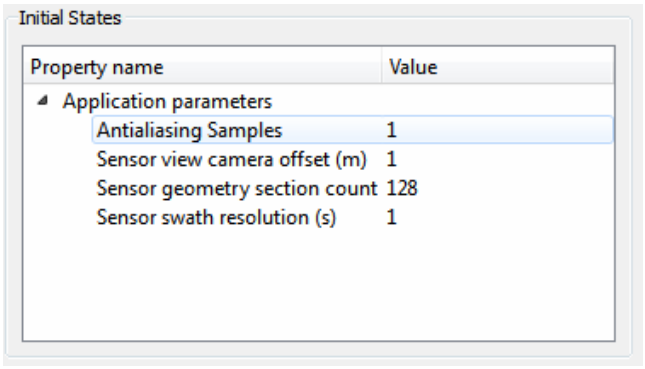
The main navigation controls in Celestia are:

- Left click + mouse move: camera pointing
- Right click + mouse move: rotation around the selected object
- Mouse wheel: zoom in/out on the selected object
- Shift + left click + mouse move: change field of view
- Left/right arrow keys: camera roll

All controls are described in Celestia's Help menu (the menu bar must be enabled in the Broker).

6.4.3. Specific application parameters in VTS

When adding Celestia as a VTS client application, some parameters can be set by clicking on the Celestia entry in the VTS project tree:



Celestia application parameters

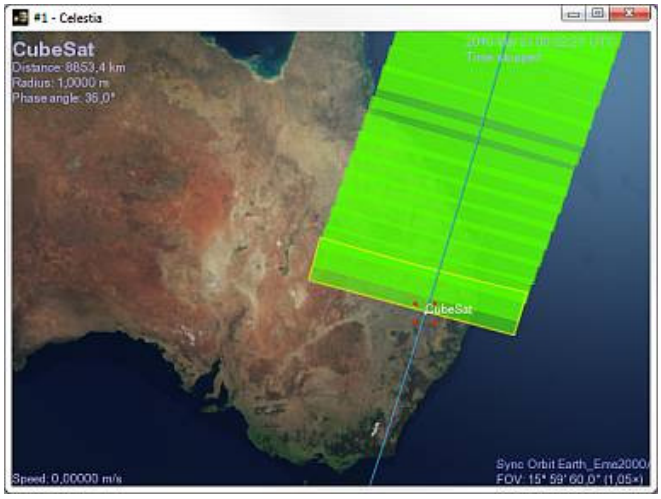
- **Antialiasing samples**: Set the level of multisample antialiasing. Not all 3D graphics hardware support antialiasing, through most newer graphics chipsets do. Larger values will result in smoother edges with a cost in rendering speed. 4 is a sensible setting for recent, higher-end graphics hardware; 2 is probably better mid-range graphics. The default value is 1, which disables antialiasing.
- **Sensor view camera offset**: The sensor camera view, accessible through 3D Camera tab in the Broker, sets the camera position at the sensor position plus an offset in meters to avoid some artifacts or obstructions of sight. This offset moves the camera along the Z axis.
- **Sensor geometry section count**: Number of points making up the outline polygon of the aiming sensor surface. Performance and display accuracy depend on this parameter.

One the one hand, performance may be affected by a too high setting and a long residual trace. On the other hand, accuracy is better with a high setting when the satellite attitude has a large angle with the nadir.

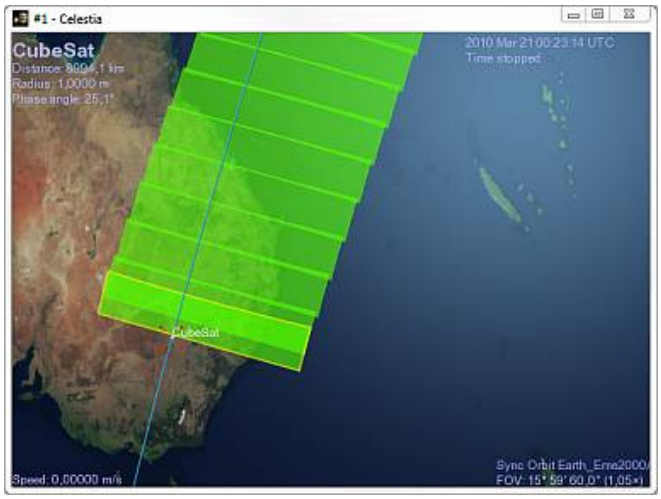
If you want to focus on sensor swath accuracy: increase the SensorGeometrySectionCount parameter and favor a short residual trace. If you want to view long missions coverage: lower the SensorGeometrySectionCount parameter with a long residual trace.

- **Sensor swath resolution**: Change the interval (in seconds) between two instantaneous sensor aiming surfaces.

Use a small value for an agile satellite, but affects the performance. Use a high value for a long coverage mission with good performance.



The default setting creates a lot of overlapping traces



A matching value between sensor width and satellite speed gives a better appearance

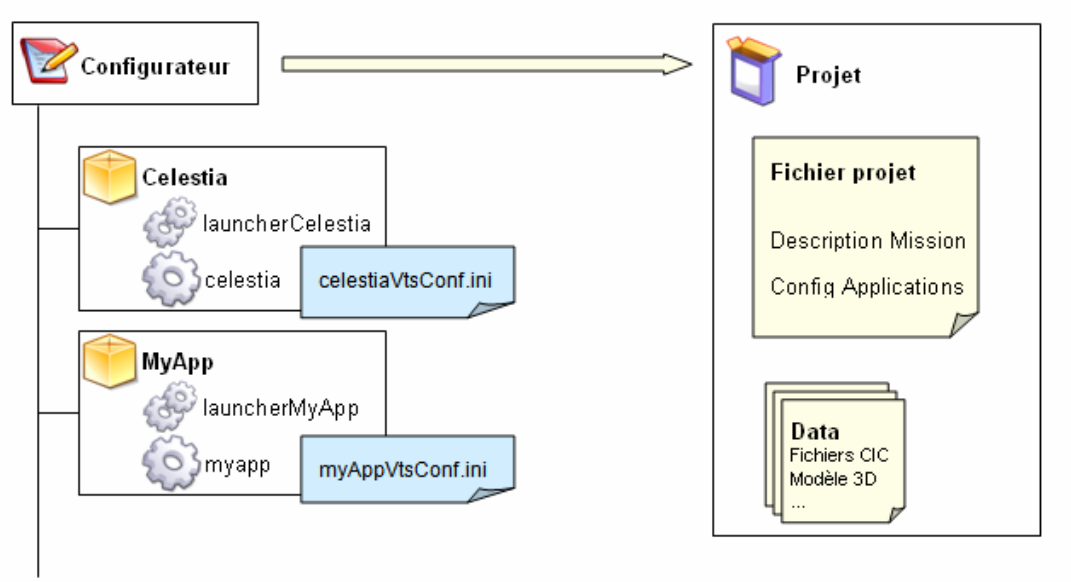
7. PLUGIN DEVELOPMENT

7.1. CLIENT APPLICATIONS IN VTS

The VTS toolkit is conceived to allow generic integration of new client applications. Client applications must provide a number of interfaces to be able to successfully connect to the toolkit. This chapter describes the architecture of the VTS toolkit and the interfaces to implement for new client applications.

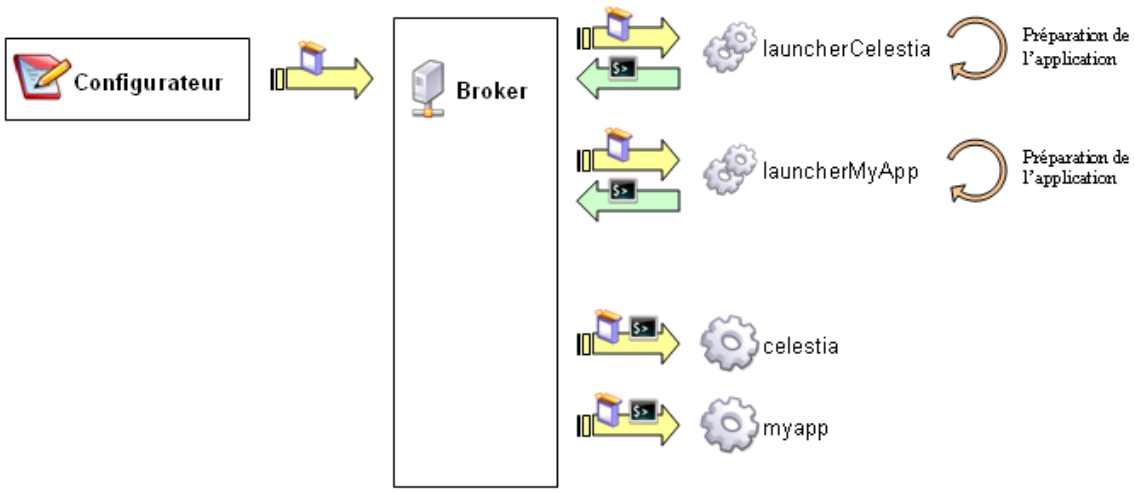
7.1.1. General architecture

A project consists in a set of entities to visualize and in the client applications launched to visualize them. Applications can be configured in the VTS configuration utility, each instance of an application separate from the others.



Project configuration

Upon startup of the visualization, the Broker starts the launchers for each instance of its client applications configured in the project file. Each launcher is passed the path to the project file and its application ID. It handles the pre-processing required to start its client application, such as data conversion, script generation, file copy, etc. It then returns to the Broker the command-line arguments to use when starting its client application. The Broker in turn starts each client applications with its provided arguments.



Visualization startup

7.1.2. Sequence

1. The Broker reads the VTS project file and builds the list of client application instances for the project.
2. The Broker executes the launchers for all client application instances.
3. The launchers pre-process the data for their application instances and return the command line for application startup to the Broker on standard ouput.
4. The Broker executes all client application instances.
5. Clients connect to the Broker on its socket and send the initialization message to indicate they are ready.
6. The Broker sends the commands for the initial states of the application declared in the *myAppVtsConf.ini* file, with default values or user-defined values from the VTS project.
7. Clients communicate with the Broker according the the synchronization protocol described in the Synchronization protocol for VTS clients chapter.

7.1.3. Application folder hierarchy and nomenclature

Client applications must follow the following folder hierarchy and nomenclature:

- The application folder must be located in the **Apps** folder of the VTS toolkit. The application folder name must start with an uppercase letter. This name will be used as the application's name.
- The application executable and its launcher must be located in the **bin** subfolder of the application folder.
- The application executable must have the same name as the application folder (case-insensitive). On a Windows platform the executable must have the *.exe* or *.bat* file extension. On a Linux platform the name can be suffixed by *.sh*, *.lnx* or might not be suffixed.

- The application launcher must be an executable and its name must start by *launcher* followed by the application name (case-insensitive).
- The optional cleaner must be an executable and its name must start by *cleaner* followed by the application name (case-insensitive), see the dedicated chapter below.
- The **doc** subfolder of the application folder may contain a **README** file. The contents of this file will be displayed in the application's data in VTS configuration tool.
- The **doc** subfolder of the application folder may contain a text file with its name starting by the application name and followed by *VtsConf.ini*. If present, this file must contain a description of the application's view properties (appearing in the view properties editor and initial properties editor for the application).
- The **doc** subfolder of the application folder may contain an icon file with *.png* or *.ico* file extension. If present, this icon will be used in the *Applications* tab of the Broker. If not present, the application executable's icon will be used instead.

Sample folder hierarchy for the Celestia client application:

- Apps
  - Celestia
    - bin
      - celestia.exe
      - launcherCelestia.exe
      - cleanerCelestia.bat
    - doc
      - README
      - celestiaVtsConf.ini
      - icon.png

7.1.4. Application launchers

A client application launcher in VTS is in charge of preparing the visualization for a client application. To this end, it may pre-process visualization data, generate scripts, copy files, etc., and lastly return the command line arguments for the client application. The only mandatory output of a launcher is the command line arguments to be passed to its client application.

7.1.4.1. Input parameters for a launcher

Launchers are all started by the Broker in a similar way, i.e. with the following command-line arguments:

- Absolute path to the project file
- Client application ID, which may be used to identify the application in the project file (useful when several instances of the same application take part in the visualization)

7.1.4.2. Preparing the client application

This phase may not be required. The launcher may need to do some processing to setup the environment for its client application. This may include altering configuration files for the application, converting project data files into application-specific file formats (e.g. CIC/CCSDS files into .xyz or .q files for Celestia), and much more.

7.1.4.3. Building the client application's command line

The launcher must provide the command line arguments for its client application. These will be appended to the application command line by the Broker. To return the command line arguments to the Broker, the launcher must print them on its standard output. Only the command line arguments may be printed to standard output. However, informational messages may be printed to standard error and will be logged by the Broker.

The first command line argument is mandatory and must be the client application ID passed by the Broker as input parameter. This ID allows the Broker to identify the client application for which the command line arguments are provided. This first field will not be passed on to the client application. Only arguments from the second field onwards will be passed.

7.1.4.4. Sample command line arguments

The launcher for the Celestia client application may print the following line to standard output:

```
0 --dir "C:/users/VTS/Apps/Celestia/bin"
  --conf "celestia.cfg"
  --url "C:/users/VTS/Apps/Celestia/bin/extras_0/VTS/CubeSat.celx"
  --extrasdir "C:/users/VTS/Apps/Celestia/bin/extras_0"
```

- The first field is mandatory and is the ID for the Celestia instance given as input to the launcher
- The following fields are the actual command line arguments for Celestia, and are standard arguments handled by the Celestia executable

The launcher for the 2DWin client application may print the following line to standard output:

```
 1 "C:/MyProject/visu.vts" 1
```

- The first field is mandatory and is the ID for the Celestia instance given as input to the launcher
- The second field is the absolute path to the project file, also given as input to the launcher, and required by the 2DWin client application
- The third field is the client application ID given as input to the launcher, and required by the 2DWin client application

The following scripts implement minimal sample launchers that directly return and empty command line argument for the client application (only the mandatory client application ID is printed):

- DOS Batch Programming version (*launcherSample.bat*):

```
@rem Input parameter %1 is the path to the VTS project file
@rem Input parameter %2 is the client application ID

@rem Mandatory client application ID output:
@echo %2

@rem End of file
```

- Shell Script version (*launcherSample.sh*):

```
#!/bin/sh
# Input parameter $1 is the path to the VTS project file
# Input parameter $2 is the client application ID
```

```
# Mandatory client application ID output:
echo $2

# End of file
```

7.1.5. Application cleaners

A Clear all client application data caches action is available in the Configurator settings menu. Each client application can provide a cleaner that removes all temporary files which could be let after many visualization runs. The executable will be started with the --clear option, as other options could appear in the future.

- Example for Celestia cleaner under Windows, launched by :

```
Apps/Celestia/bin/cleanerCelestia.bat --clear
```

- This cleaner removes all temporary folders containing 3D models and data files duplicated and generated for each run. It removes all the *extras_* temporary folders :

```
@echo off
for %%p in (%*) do (
  @rem Remove temporary folders
  if "%%p"=="--clear" (
    for /d %%a in (%~dp0/extras_*) do rd /s /q "%%a"

    @rem Return success
    exit /B 0
  )
)
```

7.2. APPLICATION IDS IN VTS

Application IDs in a VTS visualization uniquely identify any client application taking part in the visualization.

There are three classes of application IDs: project IDs, dynamic IDs, and external IDs. The distinction between the three is merely used to quickly determine the origin of an application, and is not required to be strictly followed.

- **Project IDs** are attributed in sequence by the VTS configuration utility when adding a new client application to the project. The ID is chosen as the first available (i.e. not already attributed) application ID, starting from 1.
- **Dynamic IDs** are attributed in sequence by the Broker when starting a new client application for the visualization. The ID is chosen as the first available (i.e. not already attributed) application ID, starting from 100.
- **External IDs** are attributed in sequence by the Broker upon connection of a new client, if it was not started by the Broker and requires automatic attribution of an ID (by passing the *-1* ID in the *INIT* message sent to the Broker by the client). The ID is chosen as the first available (i.e. not already attributed) application ID, starting from 1000.

7.3. SYNCHRONIZATION PROTOCOL FOR VTS CLIENTS

Once all client application launchers have successfully terminated, the Broker starts all client applications with their respective command-line arguments. Client applications must then connect to the Broker via TCP. Once connection has been established, the VTS synchronization protocol is used for communication between the Broker and its clients.

This chapter describes version 1.0 of the VTS synchronization protocol.

7.3.1. Message syntax

Messages must abide by the following syntax rules in order to be correctly parsed by VTS and standard client applications:

1. Sent messages must be followed by a terminating newline character (*\n*)
2. Message fields which contain whitespace (spaces, tabs) must be quoted using double quotes (e.g. *"field with whitespace"*)
3. Literal quotes in message fields must be escaped with a backslash (e.g. *"field with \"literal quotes\""*)
4. Literal backslashes in message fields must be escaped with an additional backslash (e.g. *"C:\\path\\to\\file"*)

7.3.2. Connecting to the Broker

In order to engage communication with the Broker, client applications must connect to a socket server setup by the Broker.

- By default, the Broker server port is 8888. In this case, the launchers are started with no specific port option.
- If VTS is configured for open ports scanning, the launchers are started with a *--serverport <portNum>* option in command line.

7.3.3. Messages received by the Broker

This section describes the commands client applications may send to the Broker.

7.3.3.1. INIT message: connection to the Broker

Once connected to the server, client applications must sent the following initialization command to engage communication:

```
INIT <Name> <ClientType> [ProtocolVersion] [ClientId]
```

The parameters for this command are described below:

Parameter	Required	Description	Format (unit)	See also
Name	Yes	Name of the client	Character string	
ClientType	Yes	Time behavior of the client	*CONSTRAINT* or *REGULATING*	Real-time VTS
ProtocolVersion	Optional (default: *1.0*)	Synchronization protocol version	Integer.Integer	
ClientId	Optional (auto attribution of an external ID if not provided)	Unique client ID	Integer	Application IDs in VTS

Commands can be sent by the client before INIT, but no TIME or DATA paquets will be received until the INIT message is sent to the Broker.

Below are some sample INIT commands:

```
INIT PrestoPlot CONSTRAINT 1.0 2
INIT Simulator REGULATING
```

7.3.3.2. TIME messages

Client applications may set the visualization date and time ratio. This is used for example when drag-and-dropping a satellite in the 2DWin client application.

TIME commands have the following syntax:

```
TIME <TimeJD1950> [TimeRatio]
```

Parameter	Required	Description	Format (unit)	See also
TimeJD1950	Yes	Visualization date in JD1950 format	Real number	Date formats in VTS
TimeRatio	Optional	Time ratio	Real number	

The date must be provided in JD1950 format. The time ratio defines the speed ratio between visualization time and wall-clock time. If it is not provided, the current time ratio is kept.

For example, the following messages set the visualization date to July 9th, 2016 at 00:00:03, with a time ratio of 10 compared to real time (10 seconds of visualization time pass with each second of wall-clock time):

```
TIME 24296.000042 10.00000
```

Only clients with CONSTRAINT time behavior may provide a time ratio. For REGULATING clients, the actual time ratio is computed by the Broker.

7.3.3.3. CMD messages

Command messages allow some level of control over the visualization. They are divided in several categories:

- *CMD TIME* commands alter the flow of time (but do not modify the visualization date or time ratio: refer to the *TIME* messages above for this)
- *CMD SERVICE* commands control various aspects of the visualization unrelated to time
- *CMD EVENT* commands deal with CIC/CCSDS mission event files for the visualization

7.3.3.3.1. CMD TIME messages

7.3.3.3.1.1. PAUSE command

The PAUSE command stops the time flow in the Broker. This is equivalent to pressing the Pause button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME PAUSE
```

When paused, the Broker keeps sending TIME messages with the current visualization date (even though it remains constant).

7.3.3.3.1.2. PLAY command

The PLAY command resumes the time flow in the Broker. This is equivalent to pressing the Play button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME PLAY
```

7.3.3.3.1.3. IncreaseTimeRatio command

The IncreaseTimeRatio command increases the time ratio in the Broker. The increase depends on the current time

ratio. The predefined time ratio sequence is: 1x, 2x, 5x, 10x, 20x, 50x, 100x, etc. This is equivalent to pressing the corresponding button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME IncreaseTimeRatio
```

7.3.3.3.1.4. DecreaseTimeRatio command

The DecreaseTimeRatio command decreases the time ratio in the Broker. It is the opposite of the IncreaseTimeRatio command. This is equivalent to pressing the corresponding button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME DecreaseTimeRatio
```

7.3.3.3.1.5. SetTimeRatio command

The SetTimeRatio command sets a custom time ratio in the Broker. It is redundant with the [TimeRatio] parameter of the TIME message, but does not require a visualization date to be provided.

This command has the following syntax:

```
CMD TIME SetTimeRatio <TimeRatio>
```

Parameter	Required	Description	Format (unit)	See also
TimeRatio	Yes	Time ratio	Real number	

For example, the following commands set a visualization time ratio 10x slower than wall-clock time:

```
CMD TIME SetTimeRatio 0.1
```

7.3.3.3.1.6. RevertTime command

The RevertTime command reverses the time flow direction. This is equivalent to pressing the corresponding button in the Broker's GUI.

This command has the following syntax:

```
CMD TIME RevertTime
```

7.3.3.3.2. CMD SERVICE messages

7.3.3.3.2.1. AUTOCLOSE command

The AUTOCLOSE command instructs the Broker to close the visualization session. All client applications are disconnected, clients started by the Broker are terminated, and the Broker itself exits.

This command has the following syntax:

```
CMD SERVICE AUTOCLOSE
```

7.3.3.3.2.2. Request replies

These commands are sent in reply of requests from the Broker. Refer to the Requests paragraph below for more information on the various Broker requests.

StoreCommand command

The StoreCommand command is sent in reply to a SaveState or SaveWindow context save request. It instructs the Broker to store a view property command for the application into the current scenario state.

This command has the following syntax:

```
CMD SERVICE StoreCommand <ViewPropertyCommand>
```

Parameter	Required	Description	Format (unit)	See also
ViewPropertyCommand	Yes	Command for setting a view property of the application	Character string	*CMD PROP* commands *CMD STRUCT* commands

Commands stored through a StoreCommand command will be sent back to the client application when the scenario state they are stored in becomes active. For more information on the project scenario, refer to the Scenario in VTS chapter.

SaveStateFinished command

The SaveStateFinished command instructs the Broker that all StoreCommand commands (see above) in reply to a SaveState or SaveWindow context save request have been sent by the client application.

This command has the following syntax:

```
CMD SERVICE SaveStateFinished
```

Synchronized command

The Synchronized command is sent by clients in reply to a SynchroRequested request, to indicate that the client is synchronized and ready for screen capture.

This command has the following syntax:

```
CMD SERVICE Synchronized
```

The recording of high quality movies in VTS requires client applications to ensure that once a Synchronized command has been issued, the visualized scene in the sender will remain frozen until the next TIME or CMD TIME PLAY message received from the Broker. Refer to the Recording movies section in the Broker user manual chapter for more information on movies in VTS.

7.3.3.3.2.3. StartApplication command

The StartApplication command instructs the Broker to start a new instance of the specified client application.

This command has the following syntax:

```
CMD SERVICE StartApplication <ApplicationName> [ApplicationId]
```

Parameter	Required	Description	Format (unit)	See also
ApplicationName	Yes	Application name	Character string	
ApplicationId	Optional (default: -1)	Application ID	Positive or null integer	Application IDs in VTS

The provided application name must match that of a client application available to VTS. If the provided application ID is already used amongst already connected clients, the clients is restarted if it is in a sleeping state. If the ID is known and the application is running, the application is not restarted.

7.3.3.3.3. CMD EVENT messages

CMD EVENT messages allow client applications to add, remove or reload CIC/CCSDS mission event files attached to project entities. The Broker will process these commands and forward them to all clients (except the sender client).

7.3.3.3.3.1. LoadFile command

The LoadFile command instructs the Broker to load a new CIC/CCSDS event file, and attach it to a project satellite.

This command has the following syntax:

```
CMD EVENT LoadFile <EventFilePath> <SatelliteFullName>
```

Parameter	Required	Description	Format (unit)	See also
EventFilePath	Yes	Absolute or project-relative path to a CIC/CCSDS event file in MEM format	Character string	
SatelliteFullName	Yes	Full name of the satellite to attach the event file to	Character string	Object paths in VTS

7.3.3.3.3.2. ReloadFile command

The ReloadFile command instructs the Broker to reload an already loaded CIC/CCSDS event file. If the specified file is not an already loaded event file, this command has no effect.

This command has the following syntax:

```
CMD EVENT ReloadFile <EventFilePath>
```

Parameter	Required	Description	Format (unit)	See also
EventFilePath	Yes	Absolute or project-relative path to a CIC/CCSDS event file in MEM format	Character string	

7.3.3.3.3.3. UnloadFile command

The UnloadFile command instructs the Broker to unload and detach an already loaded CIC/CCSDS event file from all project satellites. Events from this file are no longer attached to any satellite.

This command has the following syntax:

```
CMD EVENT UnloadFile <EventFilePath>
```

Parameter	Required	Description	Format (unit)	See also
EventFilePath	Yes	Absolute or project-relative path to a CIC/CCSDS event file in MEM format	Character string	

Note that the specified file must have been loaded previously via a LoadFile command. Event files configured in the VTS configuration utility may not be removed via the UnloadFile command: they can only be removed from the VTS configuration utility itself.

7.3.3.4. FWD messages

FWD messages provide inter-client communication, by forwarding messages from one client to another.

These messages have the following syntax:

```
FWD <RecipientId> <Message>
```

Parameter	Required	Description	Format (unit)	See also
RecipientId	Yes	Application ID of the recipient	*ALL* character string Positive or null integer	Application IDs in VTS
Message	Yes	Message to forward to the recipient	Character string (may contain spaces)	

The recipient ID must be an existing client application ID for the current visualization. If the ID is not registered in the Broker, the command is bufferised and will be send when an application connect with this exact ID.

Client application IDs are displayed in the Applications tab of the Broker. The ALL character string may be used instead of a numerical ID to broadcast the message to all clients (except the sender).

The transferred message may contain spaces. It will be transferred as is to the recipient(s).

Below are a few sample forward messages:

- Hide solar arrays in all 3D clients displaying them (the clients must implement the *Visible* structural property for the command to have any effect):

```
FWD ALL CMD STRUCT Visible "Sol/Earth/CubeSat/GS" true
```

- Reset view position and zoom level in client ID 0 (for this command to have any effect, client ID 0 must be a 2DWin client application):

```
FWD 0 CMD PROP ViewInfos Default
```

7.3.3.5. DATA messages

DATA messages are described in the Real-time VTS section below. They provide data for streamed values.

7.3.4. Common messages received by client applications

Some messages received by clients from the Broker are common across all client applications. This section describes those messages. Application-specific messages are described further below.

Messages received by client applications are categorized as follows:

- *TIME* messages concern time synchronization
- *CMD* messages control various aspects of the visualization
- *DATA* messages provide values for streamed data

Some commands have similar syntax as commands received by the Broker described in the above section.

7.3.4.1. TIME messages

Time messages synchronize client applications with the current visualization date. Client applications must handle these messages in order to properly synchronize with VTS.

When no REGULATING client is connected, the Broker broadcasts the current visualization date to all clients at a rate of 2 Hertz.

These messages have the following syntax:

```
TIME <TimeJD1950> <TimeRatio>
```

Parameters are the same as those described in the TIME messages paragraph of the Messages received by the Broker section above, except that the time ratio is always provided.

Client applications which interpolate date between time ticks sent by the Broker may use the date provided in TIME messages to synchronize their interpolation loop.

Note that when the visualization is paused, the time ratio is set to 0.

7.3.4.2. CMD messages

Command messages control various aspects of the visualization for clients. They are divided in several categories:

- *CMD TIME* commands inform about alterations to the flow of time (but do not provide the visualization date or time ratio: refer to the *TIME* messages above for this)
- *CMD SERVICE* commands control various aspects of the visualization unrelated to time

- *CMD EVENT* commands deal with CIC/CCSDS mission event files for the visualization
- *CMD CAMERA* commands define the camera position and target for client applications handling camera commands
- *CMD PROP* commands set new values for specific view properties of client applications
- *CMD STRUCT* commands set new values for structural view properties of client applications

7.3.4.2.1. CMD TIME messages

7.3.4.2.1.1. PAUSE command

The PAUSE command informs client applications that visualization is paused. Clients may then stop all animation of the visualization scene, while still letting the user interact with their GUI.

This command has the following syntax:

```
CMD TIME PAUSE
```

Visualization date is still sent by the Broker when the visualization is paused. However it remains constant, and the time ratio is set to 0.

7.3.4.2.1.2. PLAY command

The PLAY command informs client applications that visualization has resumed from pause.

This command has the following syntax:

```
CMD TIME PLAY
```

7.3.4.2.2. CMD SERVICE messages

7.3.4.2.2.1. Initialization commands

Upon connection (after reception of the INIT messages), the Broker sends several initialization commands to inform client applications of the main objects for the visualization.

InitCentralBody command

The InitCentralBody command informs client applications of a central body used in the visualization. This may be used by client applications to trigger specific initialization for this central body. One such command is sent for each central body of the visualization.

This command has the following syntax:

```
CMD SERVICE InitCentralBody <CentralBodyName> <FullParentPath>
```

Parameter	Description	Format (unit)	See also
CentralBodyName	Name of a central body	Character string	Object paths in VTS
FullParentPath	Full path to the body's parent	Character string	Object paths in VTS

InitSatellite command

The InitSatellite command informs client applications of a satellite taking part in the visualization. This may be used by client applications to trigger specific initialization for this satellite. One such command is sent for each satellite of the visualization.

This command has the following syntax:

```
CMD SERVICE InitSatellite <SatelliteName> <FullParentPath>
```

Parameter	Description	Format (unit)	See also
SatelliteName	Name of a satellite	Character string	Object paths in VTS
FullParentPath	Full path to the satellite's parent (should be a central body)	Character string	Object paths in VTS

7.3.4.2.2.2. Requests

Requests are special commands for which the Broker expects replies from client applications within a defined time frame. Refer to the Request replies paragraph above for more information on request replies.

SaveState request

The SaveState request informs client applications that a scenario state context save is ongoing, and instructs them to send commands for the view properties declared in their INI files back to the Broker.

This request has the following syntax:

```
CMD SERVICE SaveState
```

Client applications should reply using StoreCommand commands described above. Once all properties have been sent back, a SaveStateFinished command should be issued to inform the Broker that the context save is complete.

Note: Once the SaveState request has been issued by the Broker, client applications are expected to send all StoreCommand commands and the SaveStateFinished command within 3 seconds. Beyond this time frame, the context save will time out for clients which have not responded.

Refer to the Timeline toolbar section of the Scenario in VTS chapter for more information on scenario state context save actions.

Note that this request does not concern the special WindowGeometry property. This property should only be sent in reply to a SaveWindow request (see below).

SaveWindow request

The SaveWindow request informs client applications that a window position and geometry context save is ongoing, and instructs them to send back to the Broker the corresponding StoreCommand command.

This request has the following syntax:

```
CMD SERVICE SaveWindow
```

Client applications should reply with a single StoreCommand command for the specific WindowGeometry property command. The value for this property is application-specific, e.g. CMD SERVICE StoreCommand CMD PROP WindowGeometry 0 480 640 480.

Note: Once the SaveWindow request has been issued by the Broker, client applications are expected to send back a StoreCommand command for the WindowGeometry property within 1 second. Beyond this time frame, the context save will time out for clients which have not responded.

Refer to the Timeline toolbar section of the Scenario in VTS chapter for more information on window position and geometry context save actions.

SynchroRequested request

The SynchroRequested request instructs client applications that they should freeze the visualization at the last transmitted visualization date. This request is transmitted when a movie recording is ongoing in the Broker. Refer to the Recording movies section in the Broker user manual chapter for more information on movies in VTS.

This request has the following syntax:

```
CMD SERVICE SynchroRequested
```

Client applications should reply with the Synchronized command once they are synchronized with the current visualization date. The Broker will capture a movie frame once all client applications have replied (or timed out).

Note: Once the SynchroRequested request has been issued by the Broker, client applications are expected to send back a Synchronized command within 500 milliseconds. Beyond this time frame, the synchronization request will time out for clients which have not responded.

7.3.4.2.2.3. Other commands

TakeScreenshot command

The TakeScreenshot command instructs client applications to take a screenshot of the visualization scene. This may be used within scripts to save an image of the visualization at specific dates.

This command has the following syntax:

```
CMD SERVICE TakeScreenshot <Filename>
```

Parameter	Description	Format (unit)	See also
Filename	Name of the image file to save the screenshot to	Character string	

The file path must be relative to the project folder.

7.3.4.2.3. CMD EVENT messages

CMD EVENT messages instruct client applications to add, remove or reload CIC/CCSDS mission event files attached to project entities.

7.3.4.2.3.1. LoadFile command

The LoadFile command instructs client applications to load a new CIC/CCSDS event file, and attach it to a project satellite.

This command has the same syntax and parameters as the LoadFile command received by the Broker. Refer to it for details.

LoadFile commands may be sent upon connection of a new client, to instruct it to load mission event files which were loaded by other client applications before it joined the visualization.

7.3.4.2.3.2. ReloadFile command

The ReloadFile command instructs client applications to reload an already loaded CIC/CCSDS event file.

This command has the same syntax and parameters as the ReloadFile command received by the Broker. Refer to it for details.

7.3.4.2.3.3. UnloadFile command

The UnloadFile command instructs client applications to unload and detach an already loaded CIC/CCSDS event file from all project satellites. Events from this file should no longer be attached to any satellite.

This command has the same syntax and parameters as the UnloadFile command received by the Broker. Refer to it for details.

7.3.4.2.4. CMD CAMERA commands

CMD CAMERA messages instruct client applications to change the visualization camera. Only messages corresponding to cameras declared in the client application's INI file may be received.

These commands have the following syntax:

```
CMD CAMERA <CameraName> <CameraParameters>
```

The table below describes available cameras:

Camera name	Parameters (unit)	Description	See also
Synchronous	**objName**: Celestia full name of the target object (e.g.: "Sol/Earth/CubeSat_ref/CubeSat") **refObjectName**: Celestia full name of the reference frame to attach to (e.g.: "Sol/Earth/CubeSat_ref/CubeSat_Eme2000Axes", "Sol/Earth/CubeSat_ref/CubeSat_SunDir") **direction**: frame axis from which to point to the object (*X*, *-X*, *Y*, *-Y*, *Z*, *-Z* or *XYZ*) **distanceFactor** (optional): distance factor along the **direction** (relative to the default distance)	Generic camera attached to a reference frame, pointing to the target object from the given frame axis. This is the camera used by VTS for INI camera types *Body_Synchronous*, *Body_Inertial*, *Satellite_Inertial*, *Satellite_Sun*, *Satellite_SatFrame*, *Satellite_QswFrame*, and *Satellite_TnwFrame*.	
Goto	**objName**: name of the target object (e.g.: "Earth")	Go to the target object and attach to its reference frame. This is the camera used by VTS for INI camera types *Body_Goto* and *Satellite_Goto*.	
Center	**objName**: name of the target object (e.g.: "CubeSat")	Point towards the target object. The camera remains at its current location, attached to its current reference frame. This is the camera used by VTS for INI camera types *Body_Center* and *Satellite_Center*.	
CameraOrbitSat	**objName**: name of the target object (e.g.: "CubeSat")	Position the camera along the normal vector of the target object's orbital plane, pointing along that vector, so that the full orbit path of the target object can be seen. This is the camera used by VTS for INI camera type *Satellite_Orbit*.	
CameraSensorView	**sensorName**: Celestia full name of the target sensor (e.g.: "Sol/Earth/CubeSat_ref/CubeSat/Sensor_sens_ref/Sensor") **halfAngleOnX** and **halfAngleOnY** (real number, in radians): aperture half-angles of the target sensor **up** (optional, "X" or "Y"): UP vector for the camera	Position the camera from the target sensor's point of view, pointing along its aim direction. The field of view should be adjusted to match that of the sensor. This is the camera used by VTS for INI camera type *Sensor_SensorView*.	

Camera name	Parameters (unit)	Description	See also
WindowSensorView	**sensorName**: Celestia full name of the target sensor (e.g.: "Sol/Earth/CubeSat_ref/CubeSat/Sensor_sens_ref/Sensor") **halfAngleOnX** and **halfAngleOnY** (real number, in radians): aperture half-angles of the target sensor **width** (integer number, in pixels): final width of the window. Height depends on sensor ratio. **up** (optional, "X" or "Y"): UP vector for the camera	Same as CameraSensorView, except that the window is resized to fit exactly the sensor.	

Refer to the Available cameras section of the Description of application properties chapter for more information on camera capability declarations for client applications.

7.3.4.2.5. CMD PROP commands

CMD PROP commands set new values for specific properties of a client application. These properties are declared in the [SPECIFIC] section of the application's INI file.

Refer to the Description of application properties chapter for more information on application properties. These properties include for example the equatorial grid in Celestia or the zoom level and view position in 2DWin.

These commands have the following syntax:

```
CMD PROP <PropertyName> <PropertyValue>
```

Parameter	Description	Format (unit)	See also
PropertyName	Property name	Character string	
PropertyValue	Property value	Property-specific (may be composed of several space-separated fields)	Messages received by Celestia Messages received by 2DWin

CMD PROP commands are sent upon modification of specific properties in the view properties editor of the Broker, and when a scenario state becomes active and its view properties are restored.

Below are some sample CMD PROP commands:

- Enable the equatorial grid in Celestia

```
CMD PROP equatorialgrid true
```

- Reset the view position and zoom level in 2DWin

```
CMD PROP ViewInfos Default
```

7.3.4.2.6. CMD STRUCT commands

CMD STRUCT commands set new values for structural properties of a client application. These properties are declared in the entity sections of the application's INI file.

Refer to the Description of application properties chapter for more information on application properties.

These commands have the following syntax:

```
CMD STRUCT <PropertyName> <EntityFullName> <PropertyValue>
```

Parameter	Description	Format (unit)	See also
PropertyName	Property name	Character string	

Parameter	Description	Format (unit)	See also
EntityFullName	Full name of a target entity for the property	Character string	Object paths in VTS
PropertyValue	Property value	Property-specific (may be composed of several space-separated fields)	Messages received by Celestia Messages received by 2DWin

Below are some sample CMD STRUCT commands:

- Set the scale factor for Earth in Celestia

```
CMD STRUCT BodyScale "Sol/Earth" 0.8
```

- Disable the sensor footprint in 2DWin

```
CMD STRUCT AimContourVisible "Sol/Earth/CubeSat/Sensor" false
```

7.3.5. Messages received by Celestia

Apart from the common messages described above, Celestia handles the commands corresponding to properties declared in its INI configuration file.

7.3.5.1. CMD PROP commands

Property name ***Property label***	Parameters format (unit)	Description	Default value	Propagation mode	See also
AntialiasingSamples *Antialiasing Samples*	0/1/2/4/8	Antialiasing setting for Celestia.	1	INITIAL	Celestia user manual, section Specific application parameters in VTS
SensorCameraOffset *Sensor view camera offset*	Positive real number (meters)	Position offset of the sensor view camera	1	INITIAL	Celestia user manual, section Specific application parameters in VTS
SensorGeometrySectionCount *Sensor geometry section count*	Positive integer number	Number of points making up the outline polygon	128	INITIAL	Celestia user manual, section Specific application parameters in VTS
SensorSwathResolution *Sensor view camera offset*	Positive real number (s)	Interval between two instantaneous sensor aiming surfaces	1.	INITIAL	Celestia user manual, section Specific application parameters in VTS
WindowGeometry *Window geometry*	4 integers (pixels)	Window position and size (x, y, width, height)	0,0,640,480	MANUAL	
equatorialgrid *Equatorial grid*	Boolean (*false* or *true*)	Display of the celestial equatorial grid	false	MANUAL	

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
CameraDesc *Camera parameters*	Character string	Camera description (target, reference frame, position, field of view)	"Default"	MANUAL	*CMD CAMERA* commands
SelectObject *Selected object*	Character string	Selected object in the 3D view (Celestia full name)	"Sol/Earth"	MANUAL	Object paths in VTS
SatelliteLabelsVisible *Satellite labels*	Boolean (*false* or *true*)	Visibility of all satellite labels	true	MANUAL	
SolarSystemScale *Solar system scale*	Positive real number	Scale factor for the solar system	1.	MANUAL	Scale factors in VTS
WindowMenus *Window menu*	Boolean (*false* or *true*)	Display of the menu bar	false	MANUAL	
WindowText *Window text*	Boolean (*false* or *true*)	Display of text information on the current object	true	MANUAL	
AmbientLight *Ambient Light*	Real number within [0,1]	Amount of articifial ambient light	0.3	MANUAL	
AllPoiVisible *All POI visibility*	Boolean (*false* or *true*)	Display of all Points of Interest	true	MANUAL	POIs and ROIs in VTS
AllRoiVisible *All ROI visibility*	Boolean (*false* or *true*)	Display of all Regionsof Interest	true	MANUAL	POIs and ROIs in VTS

7.3.5.2. CMD STRUCT commands

7.3.5.2.1. Central body properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
BodyScale *Body scale*	Positive real number	Scale factor for the central body	1.	MANUAL	Scale factors in VTS
Eme2000AxesVisible *EME2000 inertial frame axes*	Boolean (*false* or *true*)	Display of the EME2000 inertial reference frame	false	MANUAL	
FrameAxesVisible *Body frame axes*	Boolean (*false* or *true*)	Display of the local body frame	false	MANUAL	
PlanetographicGridVisible *Planetographic grid*	Boolean (*false* or *true*)	Display of the planetographic grid	false	MANUAL	

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
TerminatorVisible *Terminator*	Boolean (*false* or *true*)	Display of the day-night terminator	false	MANUAL	

7.3.5.2.2. Satellite and subpart properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
Visible *Component visibility*	Boolean (*false* or *true*)	Visibility of the object	true	MANUAL	

7.3.5.2.3. Satellite properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
SatelliteScale *Satellite scale*	Positive real number	Scale factor for the whole satellite	1.	MANUAL	Scale factors in VTS
OrbitVisible *Orbit path*	Boolean (*false* or *true*)	Display of the satellite's orbit path	true	MANUAL	
OrbitWindow *Orbit time window*	Two Positive real numbers Duration before event, duration after event, in hours	Duration of the satellite's orbit path before and after the satellite	2 2	MANUAL	
Eme2000AxesVisible *EME2000 inertial frame axes*	Boolean (*false* or *true*)	Display of the EME2000 inertial reference frame	false	MANUAL	
QswAxesVisible *QSW local frame axes*	Boolean (*false* or *true*)	Display of the QSW local orbital frame	false	MANUAL	
TnwAxesVisible *TNW local frame axes*	Boolean (*false* or *true*)	Display of the TNW local orbital frame	false	MANUAL	
FrameAxesVisible *Satellite frame axes*	Boolean (*false* or *true*)	Display of the local satellite frame	true	MANUAL	
SunDirectionVisible *Sun direction*	Boolean (*false* or *true*)	Display of the Sun direction vector	false	MANUAL	
BodyDirectionVisible *Body direction*	Boolean (*false* or *true*)	Display of the satellite body direction vector	false	MANUAL	

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
VelocityVectorVisible *Velocity vector*	Boolean (*false* or *true*)	Display of the satellite velocity vector	false	MANUAL	

7.3.5.2.4. Sensor properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
AimContourVisible *Sensor contour*	Boolean (*false* or *true*)	Display of the sensor base contour	true	MANUAL	
AimVolumeVisible *Sensor volume*	Boolean (*false* or *true*)	Display of the sensor volume	true	MANUAL	
AimAxisVisible *Sensor axis*	Boolean (*false* or *true*)	Display of the sensor aim axis	false	MANUAL	
AimTraceVisible *Sensor swath*	Boolean (*false* or *true*)	Display of the sensor swath trace	false	MANUAL	

7.3.5.2.5. ROI properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
RoiVisible *ROI visibility*	Boolean (*false* or *true*)	Display the region of interest	true	MANUAL	POIs and ROIs in VTS

7.3.5.3. Scriptable CMD CAMERA commands

Scriptable CMD CAMERA commands are not used by the Broker, but they can be used inside VTS scripts (refer to the Scripts and macros in VTS chapter for more information).

These commands have the following syntax:

```
CMD CAMERA <CameraName> <CameraParameters>
```

The table below describes available cameras:

Camera name	Parameters (unit)	Description	See also
CreateMultiView	**X Y**: where X is the number of views horizontaly and Y the number of views verticaly	Creates multi viewports, each viewport has a scriptable camera.	
SelectView	**X Y**: coordinates of the viewport to control in MultiView mode	Select a viewport, the next command will only affct this viewpot.	
SetFov	**angle**: Field Of View angle in randians	Changes the current camera field of view	

7.3.6. Messages received by 2DWin

Apart from the common messages described above, 2DWin handles the following commands corresponding to properties declared in its INI configuration file.

7.3.6.1. CMD PROP commands

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
WindowGeometry *Window geometry*	4 integers (pixels)	Window position and size (x, y, width, height)	0,0,640,480	MANUAL	
ViewInfos *View parameters*	Character string (internal format)	View position and zoom level	"Default"	MANUAL	
AllPoiVisible *All POI visibility*	Boolean (*false* or *true*)	Display of all Points of Interest	true	MANUAL	POIs and ROIs in VTS
AllRoiVisible *All ROI visibility*	Boolean (*false* or *true*)	Display of all Regionsof Interest	true	MANUAL	POIs and ROIs in VTS
SensorGeometrySectionCount *Sensor geometry section count*	Positiveinteger number	Number of points making up the outline polygon	128	INITIAL	2dWin user manual, section Specific application parameters in VTS
SensorCoverageSignificantThreshold *Sensor coverage significant threshold*	Positive real number (%)	5.	Minimal percentage of new coverage	INITIAL	2dWin user manual, section Specific application parameters in VTS
SensorCoverageMergingThreshold *Sensor coverage merging threshold*	Positive real number (%)	90.	Minimal percentage for merging polygons	INITIAL	2dWin user manual, section Specific application parameters in VTS

7.3.6.2. CMD STRUCT commands

7.3.6.2.1. Central body properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
TerminatorVisible *Terminator*	Boolean (*false* or *true*)	Display of the day-night terminator	true	MANUAL	
SubsolarPointVisible *Subsolar point*	Boolean (*false* or *true*)	Display of the Sun zenith position	true	MANUAL	

7.3.6.2.2. Satellite properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
OrbitVisible *Orbit path*	Boolean (*false* or *true*)	Display of the satellite orbit path	true	MANUAL	
OrbitWindow *Orbit time window*	Two Positive real numbers Duration before event, duration after event, in hours	Duration of the satellite's orbit path before and after the satellite	2 2	MANUAL	
VisibleEvents *Visible events*	Character string list	List of visible event types	"*"	MANUAL	*Events* tab in the Broker user manual

7.3.6.2.3. Sensor properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
AimContourVisible *Sensor footprint*	Boolean (*false* or *true*)	Display of the sensor footprint (intersection with the central body)	true	MANUAL	
AimTraceVisible *Sensor swath*	Boolean (*false* or *true*)	Display of the sensor swath trace	true	MANUAL	

7.3.6.2.4. POI properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
PoiVisible *POI visibility*	Boolean (*false* or *true*)	Display the point of interest	true	MANUAL	POIs and ROIs in VTS

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
PoiTextVisible *Text visibility*	Boolean (*false* or *true*)	Display the label of the point of interest	true	MANUAL	POIs and ROIs in VTS

7.3.6.2.5. ROI properties

Property name *Property label*	Parameters format (unit)	Description	Default value	Propagation mode	See also
RoiVisible *ROI visibility*	Boolean (*false* or *true*)	Display the region of interest	true	MANUAL	POIs and ROIs in VTS
RoiTextVisible *Text visibility*	Boolean (*false* or *true*)	Display the label of the region of interest	true	MANUAL	POIs and ROIs in VTS

7.3.7. Real-time VTS

7.3.7.1. Time synchronization with an external time source

By default, the VTS Broker regulates the flow of visualization time for all client applications. This is the playback visualization mode: visualization data is read from CIC/CCSDS data files and time flow can be fully controlled (play/pause, time ratio, flow direction, etc.).

The real-time visualization mode allows VTS to synchronize with an external time source. Upon connection to the Broker, a client wishing to control the visualization time may declare itself as REGULATING (as described in the INIT message: connection to the Broker section above). The Broker then switches to real-time mode:

- time control buttons and commands are disabled
- visualization date is received from the regulating client

Note that several regulating clients may not connect to the Broker simultaneously.

The regulating client must send TIME messages to the Broker to set the visualization date, as described in the TIME messages section above, e.g.:

```
TIME 22105.5468754
```

To ensure good quality of synchronization for the visualization date, a TIME message should be sent every second of wall-clock time. Increasing the rate will not necessarily increase the quality of synchronization, due to communication delays and congestion which are better smoothed out by interpolation at a synchronization rate of 1Hz. Decreasing the rate is discouraged as it may lead to misleadingly inaccurate interpolation in client applications.

The VTS Broker estimates the current visualization date based on the TIME messages received from the regulating client, and relays the visualization date to all constrained clients, similarly to what happend in playback mode.

To ensure correct interpolation of streamed data, the Broker may introduce a delay between the moment it receives the current visualization date from the regulating client and the moment it relays it to constrained clients. This delay is based on the refresh rate of Stream data: it is computed as twice the maximum refresh rate of all INTERPOL mode streamed data. For example, if all streamed data values from a simulator are broadcasted every second, the Broker will introduce a 2-seconds delay in visualization time. However, if all streamed data are in DIRECT mode, no delay is introduced.

Lastly, it should be noted that it is important for the visualization date sent by the regulating client to be recent, i.e. it

must be a good representation of the regulating client's clock. The Broker uses the system time at packet reception to relate system time to regulating client time.

7.3.7.2. Supplying real-time data

Besides visualization date, data values may also be supplied by a client application. This is the purpose of DATA messages:

```
DATA <JD1950 date> <data ID> "<data value>"
```

For example:

```
DATA 20447.000174 pos "-6538.3475061863419 2703.5361504162843 197.30707005759857"
DATA 20447.000174 quat "0.22339793344197931 0.65359251975872334 0.53489502701655522
0.48661842497202529"
DATA 20447.000174 angle "0.175"
```

The sections below describe the parameters to a DATA message.

Note that previous versions of VTS required DATA messages to provide a protocol version number. Since this version number is now specified in the INIT message, they should no longer be provided with DATA messages. In order to retain compatibility with previously written clients, the Broker still handles a version number in DATA messages.

7.3.7.2.1. Data value date

The <JD1950 date> field provides the recording date of the data value. In most cases, this date is the same as the sending date of the message.

Due to the visualization date delay introduced by the Broker, clients receive data messages slightly in advance. This allows them to store the values and interpolate data for the current visualization date (as opposed to extrapolation were no future data values available).

7.3.7.2.2. Data identifier

The <data ID> field provides the unique identifier of the data. This ID is matched with the Stream ID set for streamed data in the VTS configuration utility. Refer to the Data sources in VTS chapter for further information on streamed data configuration.

7.3.7.2.3. Data value

The <data value> field provides the value for the data at the given date. The value must be quoted, no matter its dimension (scalar value, vector, etc.). The unit for the value depends on the data type, e.g.:

- Position of a satellite: kilometers
- Position of a satellite subpart: meters
- Quaternion: no unit
- Angle: degrees

7.3.8. Sample session

This example shows the messages sent by a regulating client defining a square equatorial orbit (don't try this at home!), at 100x wall-clock time:

Sending date	Message
T0	INIT example REGULATING
T0	TIME 23000.000000
T0	DATA 23000.000000 pos "10000 0 0"
T0+1s	TIME 23000.001157
T0+1s	DATA 23000.001157 pos "0 10000 0"
T0+2s	TIME 23000.002315
T0+2s	DATA 23000.002315 pos "-10000 0 0"
T0+3s	TIME 23000.003472
T0+3s	DATA 23000.003472 pos "0 -10000 0"
T0+4s	TIME 23000.004630
T0+4s	DATA 23000.004630 pos "10000 0 0"
T0+5s	TIME 23000.005787
T0+5s	DATA 23000.005787 pos "0 10000 0"
T0+6s	Etc.

7.4. DESCRIPTION OF APPLICATION PROPERTIES

The VTS toolkit relies on a command mechanism to control client applications from the Broker in a centralized way. This mechanism is used to send view properties commands upon activation of a new scenario state.

Client applications may describe view properties which will be displayed in the view properties editor of the VTS configuration toolkit or of the Broker. Initial properties sent to clients upon connection may also be described.

3D client applications may also declare compatible cameras available in the 3D Cameras tab of the Broker. It is the client application's responsibility to ensure it is compatible with the cameras it declares to have available.

7.4.1. Application properties file format

The applications properties file must be located in the doc subfolder of the application folder with the following name: <application name>VtsConf.ini

This file is in standard INI format. It must be composed of sections with uppercase names. Properties in each section are stored in associative arrays with keys prefixed by a number, in order to preserve the order of declarations when displaying the properties in VTS. Array keys must be written in lowercase.

Important: array keys in a single section are numbered from 1 to N. The section must hold the "size = N" statement.

7.4.2. Section list

- **[INITIAL]**: Initial properties for the application. These properties are configured in the application's pane in the *Structure* tab of the VTS configuration utility. The propagation mode must be set to *INITIAL* for all properties in this section.
- **[SPECIFIC]**: Specific properties for the application.

- **[BODY]**: Structural properties for central bodies.
- **[COMPONENT]**: Structural properties for satellite components. These will be available for the root component of a satellite and for all of its subparts.
- **[SATELLITE]**: Structural properties for satellites.
- **[SENSOR]**: Structural properties for satellite sensors.
- **[CAMERAS]**: Compatible cameras for the application.
- **[ROI]** and **[POI]**: Properties for regions and points of interest.

7.4.3. Property declaration

Each property must be declared with the following required fields:

- **name**: Name of the property. This name must be unique within the current INI section. It will be used as command name in the VTS synchronization protocol.
- **type**: Property data type. This may be a basic type, a Qt type, or a VTS type. See below for further details.
- **defaultValue**: Default value for the property.
- **propagation**: Propagation mode for the property. See below for further details.
- **label**: Label of the property, displayed in the view properties editor.

Below is an example of an application properties file:

```
[INITIAL]

; XMLFile
1/name = XMLFile
1/type = DataFile_t
1/defaultValue = "Data/ft.xml"
1/propagation = INITIAL
1/label = Transfer Function File

size = 1


[SPECIFIC]

; WindowGeometry
1/name = WindowGeometry
1/type = QRect
1/defaultValue = @Rect(0 0 640 480)
1/propagation = MANUAL
1/label = Window geometry

size = 1


[SATELLITE]

; SatelliteScale
1/name = SatelliteScale
1/type = SatelliteScale_t
1/defaultValue = 1.0
```

```
1/propagation = MANUAL
1/label = Satellite scale

size = 1
```

7.4.4. Available property types

There are three categories of property types: basic types, VTS types, and Qt types.

7.4.4.1. Basic types

The table below lists the available basic types:

Type	Description	Example	Editor
bool	Boolean value	true, false	Checkbox
int	Integer number	42	Text box
double	Real number	1.618	Text box

7.4.4.2. VTS types

VTS types are convenience types handled by VTS.

The table below lists the available VTS types:

Type	Description	Example	Editor
EntityScale_t	Scale factor for a celestial body (double)	0.5	Slider with a zoom factor of 1000
SatelliteScale_t	Scale factor for a satellite (double)	1000	Slider with a zoom factor of 100
EntityRange_t	Interval of real values [0,1]	0.75	Slider with min value 0, max value 1, default value 30%
DataFile_t	Relative path to a data file in the project folder	"Data/config.xml"	Text field with browse button and copy dialog if the selected file is outside the project folder
ExternalFile_t	Absolute path to a data file (portability of the project may break on other machines)	"C:/Generate/today.oem"	Text field with browse button allowing absolute paths
CameraDesc_t	Camera description for Celestia	N/A	Specific camera editor
TimeWindow_t	Time distribution before and after a dated event. Durations should be in hours.	5 1	Slider setting the total duration distribution bewteen before and after the event. Changing the "before" or "after" duration modifies the total duration but not the distribution.

7.4.4.3. Qt types

Qt types are those handled by the QVariant class from the Qt library.

The table below lists the available Qt types:

Type	Description	Example	Editor
QString	Character string	"Sol/Earth"	Text box
QStringList	List of character strings (comma-separated)	"2dWin", "Celestia"	Text box
QRect	Rectangle (integer coordinates)	@Rect(0 0 640 480)	Spin boxes (X, Y, width, height)

7.4.5. Available propagation modes

The following propagation modes are available:

- **MANUAL**: The property may be set independently for each scenario state. Its value may be propagated across states with the propagation buttons in the view properties editor.
- **AUTO**: The property has a unique value for all scenario states. Its value is automatically propagated across the whole scenario upon modification. Its name is displayed in italics in the view properties editor. This propagation mode should be used in rare cases where the property should remain consistent during the whole project, for example the visibility of a toolbar.
- **INITIAL**: The property is sent to the client application upon startup and may not be altered dynamically nor resent. This propagation mode must be used only for properties in the *INITIAL* section.

7.4.6. Available cameras

The [CAMERAS] section lists VTS cameras available in the client application. Refer to the 3D Cameras tab section in the Broker user manual chapter for a description of all VTS cameras. For each available camera, several orientations must be implemented.

```
[CAMERAS]
1/type = Body_Synchronous
2/type = Body_Inertial
size = 2
```

The following VTS cameras may be declared:

Target entity	Camera name	Corresponding Broker cameras
Central body	**Body_Synchronous**	*Fixed in Earth frame* *North pole* *South pole*
	Body_Inertial	*Inertial*
	Body_Goto	*Goto*
	Body_Center	*Center*
Satellite	**Satellite_Inertial**	*Inertial cameras*
	Satellite_Sun	*Sun and* Body *cameras* : • *View from Sun* • *View toward Sun*
	Satellite_SatFrame	*Satellite frame cameras*

Target entity	Camera name	Corresponding Broker cameras
	Satellite_QswFrame	*Sun and* Body *cameras* : • *View from* Body • *View toward* Body *QSW frame cameras*
	Satellite_TnwFrame	*TNW frame cameras*
	Satellite_Orbit	*Miscellaneous cameras* : • *Orbit*
	Satellite_Goto	*Miscellaneous cameras* : • *Goto*
	Satellite_Center	*Miscellaneous cameras* : • *Center*
Sensor	**Sensor_SensorView**	*Sensors*

Refer to the CMD CAMERA commands section in the Synchronization protocol for VTS clients chapter for more information on the corresponding camera messages sent to client applications.

7.4.7. Usage in the VTS synchronization protocol

Properties and cameras declared in the INI file are used by the VTS synchronization protocol to generate commands for client applications. Properties in the [INITIAL] and [SPECIFIC] sections are translated into PROP commands, while those in entity-specific sections ([BODY], [COMPONENT], [SATELLITE], and [SENSOR]) are translated into STRUCT commands. Cameras in the [CAMERAS] section result in corresponding CAMERA commands being sent.

Specific properties:

```
CMD PROP <PropertyName> <PropertyValue>
```

For example:

```
CMD PROP WindowGeometry 0 1 640 480
```

Structural properties:

```
CMD STRUCT <PropertyName> <EntityFullName> <PropertyValue>
```

For example:

```
CMD STRUCT AimContourVisible "Sol/Earth/CubeSat/Sensor" true
```

Cameras:

```
CMD CAMERA <CameraType> <CameraParameters>
```

Note that the CameraType parameter is not identical to the camera name listed in the [CAMERAS] section of the INI file.

For example:

```
CMD CAMERA CameraSensorView "Sol/Earth/CubeSat_ref/CubeSat/Sensor_sens_ref/Sensor"
0.872665 0.349066
```

7.5. VTS PROJECT FILE FORMAT

A VTS project file describes all the elements of a visualization. It is written in XML format for clarity. Its name must end with the .vts extension.

A sample file is provided at the end of this chapter.

7.5.1. Notation

The sections below are structured as a tree representing the XML tags contained in a VTS project file.

Section names follow this nomenclature:

- Tags written as *<Tag>* may contain other tags.
- Tags written as *<Tag/>* are self-contained (they may not contain other tags).
- Tags written as *<*Tag**>* are generic tags which are used in various contexts.

Generic tags are indicated under the sections of tags they may be child of. However, they are only fully described in separate sections outside of the main structure.

7.5.2. <Project> : Project definition

The Project tag contains all project elements. Il defines the revision number of the VTS configuration utility used to generate the file. This revision number is used to guarantee backwards-compatibility of project files.

The current format of this tag has been introduced at revision r1168. If not found, the XML reader will attempt loading the project file with an older schema.

Attribute	Description	Format (unit)	See also
Revision	Project revision	Positive integer	Backwards compatibility in VTS

Sample Project tag:

```
<Project Revision="1645">
```

7.5.2.1. <General/> : General project parameters

The General tag defines the general parameters of a project in its attributes.

Attribute	Description	Format (unit)	See also
Name	Project name	Character string	
StartDateTime	Visualization start date	• Real number (JD1950) • Integer + Real number (MJD, default) • Character string (ISO)	Date formats in VTS
EndDateTime	Visualization end date	• Real number (JD1950) • Integer + Real number (MJD, default) • Character string (ISO)	Date formats in VTS

Sample General tag:

```
<General Name="CubeSat" StartDateTime="21994.000000000"
EndDateTime="21995.975690000"/>
```

7.5.2.2. <MetaData/> : Meta information about the project

The MetaData tag shows meta information about the project in a description field.

Attribute	Description	Format (unit)	See also
Description	Project description	Character string	

Sample MetaData tag:

```
<MetaData Description="Sample project"/>
```

7.5.2.3. <StartOptions/> : Initial parameters for the visualization

The StartOptions tag defines the initial parameters for the Broker at the start of the visualization.

Attribute	Description	Format (unit)	See also
TimeRatio	Time ratio at the start of the visualization	Positive real number	Time management in the Broker user manual
SysTimeSynced	Synchronization of visualization time with system time	Boolean	
Paused	Pause time at the start of the visualization	Boolean	Time management in the Broker user manual
Looped	Loop playback for the visualization	Boolean	Time management in the Broker user manual
Minimized	Minimize the Broker window at the start of the visualization	Boolean	
Hidden	Hide the Broker window at the start of the visualization	Boolean	
AutoClosed	Automatically close the Broker once visualization has reached its end date	Boolean	

Sample StartOptions tag:

```
<StartOptions TimeRatio="100" SysTimeSynced="0" Paused="0" Looped="1" Minimized="0"
Hidden="0" AutoClosed="0"/>
```

7.5.2.4. <BrokerOptions/> : Parameters for the Broker window

The BrokerOptions tag defines the display mode of the Broker and some of its window parameters.

Attribute	Description	Format (unit)	See also
WindowMode	Window mode for the Broker (*Undocked*, *DockedOnTop* or *DockedOnBottom*)	Character string	Display modes in the Broker user manual
Collapsed	Compact mode for the Broker window (when undocked)	Boolean	Display modes in the Broker user manual
AlwaysOnTop	Broker window always on top of other windows	Boolean	Display modes in the Broker user manual
XPos	X coordinate of the Broker screen position	Integer	
YPos	Y coordinate of the Broker screen position	Integer	

Attribute	Description	Format (unit)	See also
Width	Width of the Broker window	Integer	
Height	Height of the Broker window (in full display mode)	Integer	
ActiveTab	Tab number of the default Broker tab (in full display mode)	Integer	

Sample BrokerOptions tag:

```
<BrokerOptions WindowMode="Undocked" Collapsed="1" AlwaysOnTop="1" XPos="0"
YPos="480" Width="640" Height="80" ActiveTab="0"/>
```

7.5.2.5. <TimelineOptions> : Display parameters for the project's timeline

The TimelineOptions tag defines the display parameters for the project's timeline, i.e. the display modes for data files loaded in the timeline in the VTS configuration utility or Broker.

This tag has no attributes.

Refer to the Timeline section in the Scenario in VTS chapter for more information.

Sample TimelineOptions tag:

```
<TimelineOptions>
```

7.5.2.5.1. <TimelineScenario/> : Scenario display parameters in the timeline

The TimelineScenario tag defines the display scenario parameters in the timeline. There can be only a single such tag in the TimelineScenario section.

Attribute	Description	Format (unit)	See also
Name	Must be "Scenario"	Character string	
Pos	Row position in the timeline	Positive integer	
Size	Row height in the timeline	Pixels	

Sample TimelineScenario tag:

```
<TimelineScenario Name="Scenario" Pos="0" Size="23"/>
```

7.5.2.5.2. <TimelineEvents/> : Events for a satellite display parameters in the timeline

The TimelineEvents tag defines the events for a satellite display parameters in the timeline. This row is automatically filled with existing events attached to a satellite.

Attribute	Description	Format (unit)	See also
Name	Must be "Events for <existing satellite name>"	Character string	
Pos	Row position in the timeline	Positive integer	
Size	Row height in the timeline	Pixels	

Sample TimelineEvents tag:

```
<TimelineEvents Name="Events for CubeSat" Pos="1" Size="23"/>
```

7.5.2.5.3. <TimelineFile/> : Display parameters of a file in the timeline

The TimelineFile tag defines the display parameters of a CIC/CCSDS file in the project's timeline.

Attribute	Description	Format (unit)	See also
Name	Name of the CIC/CCSDS file	Character string	
Pos	Row position in the timeline	Positive integer	
Size	Row height in the timeline	Pixels	
Mode	Display mode of the file in the timeline (*DEFAULT* for auto selected mode or : *Normal*, *Block*, *Event*, *Interval*, *Gradient*, *Color*, *Graph*)	Character string	Graphical representation of files in Scenario in VTS
Overlay	Flag to use an associated color file as a color overlay in the Timeline	Boolean	

The file path must be relative to the project folder.

The specified display mode must be compatible with the contents of the CIC/CCSDS file. Otherwise, it will not be applied.

Sample TimelineFile tag:

```
<TimelineFile Name="Data/CIC-Sat_SOLAR_ARRAY_CURRENT.TXT" Pos="1" Size="23"
Mode="DEFAULT" Overlay="false"/>
```

7.5.2.5.4. <TimelineScript/> : Display parameters of a script file in the timeline

The TimelineScript tag defines the display parameters of a script file in the project's timeline.

Attribute	Description	Format (unit)	See also
Name	Name of the script file	Character string	Scripts and macros in VTS
Pos	Row position in the timeline	Positive integer	
Size	Row height in the timeline	Pixels	

The file path must be relative to the project folder.

Sample TimelineScript tag:

```
<TimelineScript Name="Data/SCRIPT.TXT" Pos="3" Size="23"/>
```

7.5.2.5.5. <TimelineStream/> : Stream display parameters in the timeline

The TimelineStream tag defines a stream display parameters in the timeline.

Attribute	Description	Format (unit)	See also
Name	Must be "Stream <existing stream ID>"	Character string	
Pos	Row position in the timeline	Positive integer	
Size	Row height in the timeline	Pixels	

Sample TimelineStream tag:

```
<TimelineStream Name="Stream &lt;i&gt;Pos&lt;/i&gt;" Pos="4" Size="23"/>
```

Note : the escaped character sequence represents an italic tag.

7.5.2.6. <ToBeUsedApps> : Client applications for the visualization

The ToBeUsedApps tag lists all project client applications which are launched at the start of the visualization.

This tag has no attributes.

Refer to the VTS Broker and Client applications in VTS chapters for further information.

Sample ToBeUsedApps tag:

```
<ToBeUsedApps>
```

7.5.2.6.1. <Application> : Project client application

The Application tag defines a client application for the visualization.

Attribute	Description	Format (unit)	See also
Name	Application name	Character string	Application launchers
Id	Unique application identifier	Positive integer	Application IDs in VTS

The application name is used by the VTS Broker to build the path to the application's executable.

The unique application ID is used by the VTS Broker to identify clients upon connection and during the visualization.

Sample Application tag:

```
<Application Name="Celestia" Id="1">
```

7.5.2.6.1.1. <SpecificArgs/> : Additional parameters for an application

The SpecificArgs tag defines additional command-line parameters for an application. These parameters are automatically passed to client applications upon startup.

Attribute	Description	Format (unit)	See also
Args	Parameters list	Character string	Application launchers

The parameters list is stored as is in the project file.

Sample SpecificArgs tag:

```
<SpecificArgs Args="--bds BDS_PHR1A.tcl"/>
```

7.5.2.7. <Entities> : Entities of the visualization

The Entities tag contains all entities of the visualization: satellites, ground stations, sensors, etc.

This tag has no attributes.

Sample Entities tag:

```
<Entities>
```

7.5.2.7.1. <Body> : Central body

The Body tag defines a central body for the visualization.

Attribute	Description	Format (unit)	See also
Name	Central body name	Character string	Central bodies in VTS
ParentPath	Path to the body's parent	Character string	Object paths in VTS

Central bodies are VTS entities around which visualized satellites orbit. When using standard VTS client applications such as Celestia or 2dWin, a central body must reference a known object from VTS's default list, or provide the data required for its display, position and orientation.

Refer to the Central bodies in VTS chapter for more information on central bodies.

Sample Body tag:

```
<Body Name="Earth" ParentPath="Sol">
```

7.5.2.7.1.1. <Texture> : Body texture

The Texture tag contains the tags defining the texture of a central body.

This tag has no attributes.

Sample Texture tag:

```
<Texture>
```

This tag is empty if the central body uses its built-in texture. For further information on texture types, refer to the Texture of a body section of the VTS configuration utility user manual chapter.

<TextureFixedFile> : Fixed body texture file

The TextureFixedFile tag defines the fixed texture file for a central body.

Attribute	Description	Format (unit)	See also
Name	Fixed texture file name	Character string	Texture of a body in the VTS configuration utility user manual

Sample TextureFixedFile tag:

```
<TextureFixedFile Name="Textures/earth-AE8.png">
```

The file name must point to an image file, and its path must be relative to the project folder.

<TextureTimedFile> : Timed body texture file

The TextureTimedFile tag defines the timed texture file for a central body.

Attribute	Description	Format (unit)	See also
Name	Timed texture file name	Character string	Texture of a body in the VTS configuration utility user manual

Sample TextureTimedFile tag:

```
<TextureTimedFile Name="Data/EARTH_TEXTURE.TXT">
```

The file name must point to a CIC/CCSDS file in MEM format defining the textures to use, and its path must be relative to the project folder.

<TextureTimedFileCyclic> : Timed cyclic body texture file

The TextureTimedFileCyclic tag defines the timed cyclic texture file of a central body.

Attribute	Description	Format (unit)	See also
Name	Timed cyclic texture file name	Character string	Texture of a body in the VTS configuration utility user manual
Epoch	Texture cycle epoch	Real number (JD1950)	

Attribute	Description	Format (unit)	See also
Period	Texture cycle period	Real number (days)	

Sample TextureTimedFileCyclic tag:

```
<TextureTimedFileCyclic Name="Data/EARTH_TEXTURE_CYCLIC.TXT" Epoch="18262"
Period="2">
```

The file name must point to a CIC/CCSDS file in MEM format defining the textures to use, and its path must be relative to the project folder.

*7.5.2.7.1.2. <*Prop2d*> : 2D properties of a body*

The Prop2d tag defines the 2D properties of a central body, i.e. the icon displayed in the VTS configuration utility and for projections in 2DWin.

If this tag is not available and the central body is a standard VTS central body, then its default 2D properties are used.

Refer to the documentation of the <*Prop2d*> generic tag in the <*Prop2d*> : 2D properties section for further information.

*7.5.2.7.1.3. <*Graphics3d*> : 3D graphic properties of a body*

The Graphics3d tag defines the 3D graphic properties of a central body, i.e. its 3D model, radius, etc.

If this tag is not available and the central body is a standard VTS central body, then its default 3D graphic properties are used.

Refer to the documentation of the <*Graphics3d*> generic tag in the <*Graphics3d*> : 3D graphic properties section for further information.

7.5.2.7.1.4. <EphemerisMode> : Ephemeris mode of a body

The EphemerisMode tag defines the ephemeris mode of a body.

Attribute	Description	Format (unit)	See also
Mode	Ephemeris mode enum ("Default", "Catalog", or "User")	Character string	See the Central bodies in VTS for more informations about the different modes.

*7.5.2.7.1.5. <*Geometry*> : Geometric properties of a body*

The Geometry tag defines the position and orientation of a central body.

If this tag is not available and the central body is a standard VTS central body, then the default ephemerides are used.

Refer to the documentation of the <*Geometry*> tag in the Central bodies in VTS and <*Geometry*> : Geometric properties section for further information.

Position data for a body is expressed in kilometers.

7.5.2.7.1.6. <GroupGroundStations> : Ground stations of a body

The GroupGroundStations tag contains the list of all ground stations on a central body.

This tag has no attributes.

Sample GroupGroundStations tag:

```
<GroupGroundStations>
```

<GroundStation> : Ground station

The GroundStation tag defines a ground station of a central body.

Attribute	Description	Format (unit)	See also
Name	Ground station name	Character string	

Sample GroundStation tag:

```
<GroundStation Name="Toulouse">
```

<*Prop2d*> : 2D properties of a station

The Prop2d tag defines the 2D properties of a ground station, i.e. its icon 2D views.

Refer to the documentation of the <*Prop2d*> generic tag in the <*Prop2d*> : 2D properties section for further information.

<LatLongAlt> : Coordinates of a station

The LatLongAlt tag defines the geographical coordinates of a ground station: latitude, longitude and altitude on its central body.

This tag has no attributes.

Sample LatLongAlt tag:

```
<LatLongAlt>
```

<*Value*> : Coordinates value of a station

The Value tag defines the value of the coordinates of a ground station on its central body.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

This tag may only contain a Fixed tag: the ground station coordinates are fixed. The Fixed tag must contain a character string defining the three real number values for the latitude, longitude, and altitude of the ground station.

Latitude and longitude are expressed in degrees, altitude in meters.

Sample Value tag for ground station coordinates:

```
<Value>
   <Fixed Data="1.22 43.38 0"/>
</Value>
```

<SensorStation> : Station sensor

The SensorStation tag defines a ground station sensor. A ground station may only contain a single SensorStation tag.

This tag has no attributes.

Sample SensorStation tag:

```
<SensorStation>
```

<SensorTarget> : Target of a station sensor

The SensorTarget tag defines the target of a ground station sensor. It must contain only a single TargetAltitude or TargetEntity tag.

This tag has no attributes.

Sample SensorTarget tag:

```
<SensorTarget>
```

<TargetAltitude/> : Target altitude of a station sensor

The TargetAltitude tag defines a fixed altitude as the target of a ground station sensor.

Attribute	Description	Format (unit)	See also
Altitude	Target altitude	Positive real number (km)	

Sample TargetAltitude tag:

```
<TargetAltitude Altitude="1000"/>
```

<TargetEntity/> : Target satellite of a station sensor

The TargetEntity tag defines a satellite's altitude as the target of a ground station sensor.

Attribute	Description	Format (unit)	See also
Name	Full name of the target satellite	Character string	Object paths in VTS

Sample TargetEntity tag:

```
<TargetEntity Name="Sol/Earth/CubeSat"/>
```

<*Sensor*> : Station sensor properties

The Sensor tag defines the properties of a ground station sensor, used for displaying the sensor's footprint projected on its central body in 2DWin.

Refer to the documentation of the <*Sensor*> generic tag in the <*Sensor*> : Sensor section for further information.

7.5.2.7.1.7. <GroupPointsOfInterest> : Points of interest of a body

The GroupPointsOfInterest tag contains the list of all points of interest on a central body.

This tag has no attributes.

Sample GroupPointsOfInterest tag:

```
<GroupPointsOfInterest>
```

<PointOfInterest> : Point of interest

The PointOfInterest tag defines a point of interest (POI) of a central body.

Attribute	Description	Format (unit)	See also
Name	POI name	Character string	

Sample PointOfInterest tag:

```
<PointOfInterest Name="Landing site">
```

<CoordinatesFile> : coordinates file of a POI

The CoordinatesFile tag contains the coordinates file name of a POI.

Attribute	Description	Format (unit)	See also
Name	Name of the CIC/CCSDS file	Character string	

<Graphics> : graphics properties of a POI

The Graphics tag contains the graphics properties of a POI.

Attribute	Description	Format (unit)	See also
Color	Color of the point of interest	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	

Attribute	Description	Format (unit)	See also
Opacity	Opacity of the point of interest	Integer in the [0,100] range (percentage)	
PointSize	Size of the POI (DOT, SMALL, MEDIUM, BIG)	Integer in the [0,3] range (no unit)	

Sample PointOfInterest tag:

```
<PointOfInterest Name="Madagasikara">
 <CoordinatesFile Name="Madagasikara.txt"/>
 <Graphics Color="0 0.313832 1" Opacity="100" PointSize="2"/>
</PointOfInterest>
```

7.5.2.7.1.8. <GroupRegionsOfInterest> : Regions of interest of a body

The GroupRegionsOfInterest tag contains the list of all regions of interest on a central body.

This tag has no attributes.

Sample GroupRegionsOfInterest tag:

```
<GroupRegionsOfInterest>
```

<RegionOfInterest> : Ground station

The RegionOfInterest tag defines a region of interest (ROI) of a central body.

Attribute	Description	Format (unit)	See also
Name	ROI name	Character string	

Sample RegionOfInterest tag:

```
<RegionOfInterest Name="Target">
```

<CoordinatesFile> : coordinates file of a ROI

The CoordinatesFile tag contains the coordinates file name of a ROI.

Attribute	Description	Format (unit)	See also
Name	Name of the CIC/CCSDS file	Character string	

<Graphics> : graphics properties of a ROI

The Graphics tag contains the graphics properties of a ROI.

Attribute	Description	Format (unit)	See also
Color	Color of the ROI	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	
FillOpacity	Opacity of the ROI surface	Integer in the [0,100] range (percentage)	

Sample RegionOfInterest tag:

```
 <RegionOfInterest Name="Corsica">
  <CoordinatesFile Name="Corsica.txt"/>
  <Graphics Color="0.363836 0 1" FillOpacity="40"/>
 </RegionOfInterest>
```

7.5.2.7.2. <Satellite> : Satellite

The Satellite tag defines a satellite for the visualization.

Attribute	Description	Format (unit)	See also
Name	Name of the satellite	Character string	Structure of satellites in VTS
ParentPath	Full path to the satellite's parent	Character string	Object paths in VTS

Satellites are the main visualization entities in VTS. This tag contains all tags related to the definition of a satellite: subparts, graphic properties, etc. Refer to the Structure of satellites in VTS chapter for more information.

Sample Satellite tag:

```
<Satellite Name="SMOS" ParentPath="Sol/Earth">
```

*7.5.2.7.2.1. <*Prop2d*> : 2D properties of a satellite*

The Prop2d tag defines the 2D properties of a satellite, i.e. the icon displayed in the VTS configuration utility and for projections in 2DWin.

Refer to the documentation of the <*Prop2d*> generic tag in the <*Prop2d*> : 2D properties section for further information.

7.5.2.7.2.2. <CommonProp> : Properties of a satellite

The CommonProp tag contains property tags for a satellite.

This tag has no attributes.

Sample CommonProp tag:

```
<CommonProp>
```

<OrbitPath/> : Satellite orbit path

The OrbitPath tag defines the appearance of a satellite's orbit path, i.e. the line of its past and future locations.

Attribute	Description	Format (unit)	See also
Color	Orbit path color	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	

The satellite's orbit path can be displayed in client applications of the visualization. The path length is defined as a dynamic property available in the scenario.

Sample OrbitPath tag:

```
<Color="0.564706 0 0.00784314"/>
```

7.5.2.7.2.3. <Component> : Satellite component

The Component tag defines a satellite component. Some properties of the satellite itself are described in its top-level component (not visible in the GUI).

This tag is recursive, i.e. it may contain any number of children Component tags, which may themselves contain more Component tags. This allows describing an arbitrarily complex satellite structure. Refer to the Structure of satellites in VTS chapter for more information.

Attribute	Description	Format (unit)	See also
Name	Component name	Character string	Structure of satellites in VTS

The top-level component's name is not used: the satellite's name is used instead.

Sample Component tag:

```
<Component Name="GS">
```

<*Graphics3d*> : 3D graphic properties of a component

The Graphics3d tag defines the 3D graphic properties of a satellite component, i.e. its 3D model, radius, etc.

Refer to the documentation of the <*Graphics3d*> generic tag in the <*Graphics3d*> : 3D graphic properties section for further information.

<*Geometry*> : Geometric properties of a component

The Geometry tag defines the position and orientation of a satellite component.

Refer to the documentation of the <*Geometry*> tag in the <*Geometry*> : Geometric properties section for further information.

Position data for a satellite component is expressed in kilometers for the top-level component, and meters for its sub-components.

<SensorSatellite> : Satellite sensor

The SensorSatellite tag defines a satellite sensor.

This tag has no attributes.

Sample SensorSatellite tag:

```
<SensorSatellite>
```

<*Sensor*> : Satellite sensor properties

The Sensor tag defines the properties of a satellite sensor, used for displaying the sensor's footprint on its central body in 2DWin, and its volume in Celestia.

Refer to the documentation of the <*Sensor*> generic tag in the <*Sensor*> : Sensor section for further information.

*7.5.2.7.2.4. <*Events*> : Mission events of a satellite*

The Events tag defines the mission events for a satellite, i.e. the CIC/CCSDS event files and the custom event decorations for the satellite.

Refer to the documentation of the <*Events*> generic tag in the <*Events*> : Events section for further information.

This tag may contain both File and Decoration tags.

7.5.2.8. <*Events*> : Event decorations for all entities

The Events tag defines the default event decorations for all entities of the project. These decorations may be overridden on a per-entity basis, e.g. through the Events tag as described in the <*Events*> : Mission events of a satellite section above.

Refer to the documentation of the <*Events*> generic tag in the <*Events*> : Events section for further information.

This tag may only contain Decoration tags. It may not contain File tags, since events must be attached to an entity.

7.5.2.9. <IgnoredFiles> : Ignored files in the project timeline

The IgnoredFiles tag contains a list of CIC/CCSDS project files which should not be displayed in the project timeline.

This tag has no attributes.

Sample IgnoredFiles tag:

```
<IgnoredFiles>
```

7.5.2.9.1. <*File*> : Ignored project file in the timeline

The File tag defines a CIC/CCSDS project file which should not be displayed in the project timeline. Only files referenced by the project inside <Value> tags may be specified here.

Refer to the documentation of the <*File*> generic tag in the <*File*> : CIC/CCSDS file section for further information.

7.5.2.10. <AdditionalFiles> : Additional files in the project timeline

The AdditionalFiles tag contains a list of CIC/CCSDS files to display in the project timeline.

This tag has no attributes.

Sample AdditionalFiles tag:

```
<AdditionalFiles>
```

7.5.2.10.1. <*File*> : Additional file in the timeline

The File tag defines a CIC/CCSDS file to display in the project timeline. These files should not be referenced by the project inside <Value> tags (since files referenced by the project are automatically displayed in the project timeline -- unless ignored).

Refer to the documentation of the <*File*> generic tag in the <*File*> : CIC/CCSDS file section for further information.

7.5.2.11. <States> : Project scenario

The States tag contains a list of visualization states which compose the project scenario.

This tag has no attributes.

If present, the States tag must contain at least one Instant tag (see below) at date 0.

Refer to the Scenario in VTS chapter for more information.

Sample States tag:

```
<States>
```

7.5.2.11.1. <Instant> : Scenario visualization state

The Instant tag defines a visualization state from the project scenario. It aggregates view properties for all client applications of the visualization.

Attribute	Description	Format (unit)	See also
Time	Date of the visualization state	Real number (JD1950)	Date formats in VTS
Label	Visualization state label in the project timeline	Character string	

All projects using the project scenario must define at least one visualization state at date 0.. This state holds the values for all client application initial properties, and view properties at the start of the visualization. Refer to the Client applications in VTS chapter for more information on initial and view properties of a client application.

Sample Instant tag:

```
<Instant Time="21994.513" Label="Start of apogee thrust">
```

7.5.2.11.1.1. <AppState> : State properties for a client application

The AppState tag contains a list of properties for a specific client application. Each Instant tag must contain as many AppState tags as there are client applications in the project.

Attribute	Description	Format (unit)	See also
Id	Client application ID	Positive integer	Application IDs in VTS

The application ID must match the one defined in corresponding Application tag (see above).

Sample AppState tag:

```
<AppState Id="1">
```

<Command/> : Application property

The Command tag defines a property command for a client application. This command sets the value of an application property.

Attribute	Description	Format (unit)	See also
Str	Property command	Character string	Synchronization protocol for VTS clients

The actual value of the command depends on the client application it is intended for, and on the property to be set. Refer to the Client applications in VTS and Synchronization protocol for VTS clients chapters for more information.

Sample Command tag:

```
<Command Str="CMD PROP equatorialgrid false"/>
```

7.5.3. Generic tags

7.5.3.1. <*Sensor*> : Sensor

The Sensor tag defines a sensor, either attached to a ground station or to a satellite component.

Attribute	Description	Format (unit)	See also
Name	Sensor name	Character string	Sensors in VTS

The sensor name identifies the sensor. It is not used for a ground station sensor.

Sample Sensor tag:

```
<Sensor Name="Panchromatic">
```

7.5.3.1.1. <SensorProp> : Sensor properties

The SensorProp tag contains the properties of a sensor, both physical and graphic.

This tag has no attributes.

Sample SensorProp tag:

```
<SensorProp>
```

7.5.3.1.1.1. <SensorAttributes/> : Physical properties of a sensor

The SensorAttributes tag defines the physical properties of a sensor, i.e. its shape and aperture angles.

Attribute	Description	Format (unit)	See also
SensorType	Sensor shape	Character string (*ELLIPTICAL* or *RECTANGULAR*)	Sensors in VTS
HalfAngleX	Aperture half-angle around X	Real number	

Attribute	Description	Format (unit)	See also
HalfAngleY	Aperture half-angle around Y	Real number	

Refer to the Sensors in VTS chapter for more information.

Sample SensorAttributes tag:

```
<SensorAttributes SensorType="ELLIPTICAL" HalfAngleX="0.174533"
HalfAngleY="0.174533"/>
```

7.5.3.1.1.2. <SensorGraphics> : Graphic properties of a sensor

The SensorGraphics tag defines the graphic properties of a sensor, i.e. its appearance in client applications.

Attribute	Description	Format (unit)	See also
Range	Maximum display distance of the sensor volume in 3D	Real number (kilometers)	
VolumeColor	Color of the sensor volume in 3D	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	
VolumeOpacity	Opacity of the sensor volume in 3D	Integer in the [0,100] range (percentage)	
ContourColor	Color of the sensor footprint in 2D and 3D	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	

Only the ContourColor attribute is used for a ground station sensor. Other attributes must be present, but their values are unused.

Refer to the Sensors in VTS chapter for more information.

Sample SensorGraphics tag:

```
<SensorGraphics Range="10000"
   VolumeColor="0 1 0.603326" VolumeOpacity="60"
   ContourColor="0 1 0.603326">
```

<SensorTrace> : Graphic properties of the sensor swath trace

The SensorTrace tag defines graphic properties specific to the sensor swath trace for both 2D and 3D applications. This tag may not be present under a ground station sensor.

Attribute	Description	Format (unit)	See also
Duration	Duration of the sensor swath trace	Real number (hours)	
Opacity	Opacity of the sensor swath	Integer in the [0,100] range (percentage)	

A null trace duration displays the instantaneous sensor swath in client applications (without residual display).

The SensorTrace tag may contain no child tag, or a single FixedColor tag or ColorFile tag. If empty, the sensor swath color is defined as the satellite's orbit path color (be it a fixed color or the color from a color file attached to a Position value). This is the Orbit path color sensor swath color mode in the VTS configuration utility. Refer to the Graphic properties of a satellite sensor section in the VTS configuration utility user manual chapter for more information.

Sample SensorTrace tag:

```
<SensorTrace Duration="2" Opacity="60">
```

<FixedColor/> : Fixed color for the sensor swath

The FixedColor tag defines the fixed color of the sensor swath in client applications.

Attribute	Description	Format (unit)	See also
Value	Color for the sensor swath	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	

If present, this tag selects the Fixed color sensor swath color mode in the VTS configuration utility. Refer to the Graphic properties of a satellite sensor section in the VTS configuration utility user manual chapter for more information.

Sample FixedColor tag:

```
<FixedColor Value="1 0.0890059 0"/>
```

<*ColorFile*> : Color file for the sensor swath

The ColorFile tag defines a CIC/CCSDS color file for the color of the sensor swath in client applications.

Refer to the documentation of the <*ColorFile*> generic tag in the <*ColorFile*> : CIC/CCSDS color file section for further information.

If present, this tag selects the Color file sensor swath color mode in the VTS configuration utility. Refer to the Graphic properties of a satellite sensor section in the VTS configuration utility user manual chapter for more information.

7.5.3.1.2. <*Geometry*> : Geometric properties of a sensor

The Geometry tag defines the position and orientation of a sensor.

Ground station sensors define this tag but do not use it.

Refer to the documentation of the <*Geometry*> tag in the <*Geometry*> : Geometric properties section for further information.

Position data for a sensor is expressed in meters.

7.5.3.2. <*Prop2d*> : 2D properties

The Prop2d tag contains all tags related to the 2D properties of a visualization object.

This tag has no attributes.

Sample Prop2d tag:

```
<Prop2d>
```

7.5.3.2.1. <SymbolFile/> : 2D symbol file

The SymbolFile defines the image file to use as the icon of an object in 2D views. VTS usually provides a default icon for its various visualization entities, which can be overridden by this tag.

Attribute	Description	Format (unit)	See also
Name	2D symbol file name	Character string	2dWin user manual

The file must be in bmp, jpg or png format. The image will be scaled down to icon size (32x32) before use in 2DWin. Refer to the 2dWin user manual chapter for more information.

If the file name is empty, the default icon is used for the object.

The file name must be relative to the project folder.

Sample SymbolFile tag:

```
<SymbolFile Name="Images/ISS.jpg"/>
```

7.5.3.3. <*Graphics3d*> : 3D graphic properties

The Graphics3d tag contains all tags related to the 3D graphic properties of a visualization object.

This tag has no attributes.

Sample Graphics3d tag:

```
<Graphics3d>
```

7.5.3.3.1. <File3ds/> : 3DS file

The File3ds tag defines the 3D file in 3DS format containing the 3D model of an object, for use in 3D views.

Attribute	Description	Format (unit)	See also
Name	3DS file name	Character string	3D file format in VTS

The file must be in 3ds format.

The file name must be relative to the project folder.

Sample File3ds tag:

```
<File3ds Name="Models/ida/ida.3ds"/>
```

7.5.3.3.2. <Radius/> : Radius of an object

The Radius tag defines the bounding sphere radius of the 3D model of an object.

Attribute	Description	Format (unit)	See also
Value	Object radius	Real (kilometers or meters)	3D file format in VTS

If attached to a central body, the radius is expressed in kilometers. If attached to a satellite or component, the radius is expressed in meters.

Refer to the 3D file format in VTS chapter, and to the 3D properties of a satellite section of the VTS configuration utility user manual chapter for more information.

Sample Radius tag:

```
<Radius Value="2.79382"/>
```

7.5.3.3.3. <LightSensitive/> : Light sensitivity of an object

The LightSensitive tag defines whether or not to shade an object in 3D views.

Attribute	Description	Format (unit)	See also
Value	Object light sensitivity	Boolean	

A value of 1 enables object shading. A value of 0 disables object shading.

Refer to the 3D properties of a satellite section of the VTS configuration utility user manual chapter for more information.

Sample LightSensitive tag:

```
<LightSensitive Value="1"/>
```

7.5.3.3.4. <Use3dsCoords/> : Usage of an object's 3DS coordinates

The Use3dsCoords tag defines whether or not to use the coordinates system from the 3DS file of an object in 3D

views.

Attribute	Description	Format (unit)	See also
Value	Usage of 3DS coordinates	Boolean	3D file format in VTS
MeshScale	Scaling factor for 3DS coordinates	Real number (no unit)	

A value of 1 enables usage of 3DS coordinates. A value of 0 disables usage of 3DS coordinates.

The scale factor applies to the default mesh coordinates unit, the meter. Thus:

- A value of *1* defines coordinates in meters
- A value of *0.001* defines coordinates in millimeters
- A value of *1000* defines coordinates in kilometers

Refer to the 3D file format in VTS chapter, and to the 3D properties of a satellite section of the VTS configuration utility user manual chapter for more information.

Sample Use3dsCoords tag:

```
<Use3dsCoords Value="1" MeshScale="0.1"/>
```

7.5.3.3.5. <RotationCenter/> : Rotation center

The RotationCenter tag defines the position of the rotation center of an object.

Attribute	Description	Format (unit)	See also
X	X coordinate of the rotation center	Real number	Orientation of objects in VTS
Y	Y coordinate of the rotation center	Real number	
Z	Z coordinate of the rotation center	Real number	

Refer to the Rotation center section of the Orientation of objects in VTS chapter for more information.

Sample RotationCenter tag:

```
<RotationCenter X="2.252" Y="0" Z="0.649"/>
```

7.5.3.4. <*Geometry*> : Geometric properties

The Geometry tag defines the geometric properties of an object, i.e. its position and orientation.

This tag has no attributes.

Sample Geometry tag:

```
<Geometry>
```

7.5.3.4.1. <Position> : Position of an object

The Position tag defines the position of an object.

This tag has no attributes.

It must contain a Value tag.

Refer to the Position of objects in VTS chapter for more information.

Sample Position tag:

```
<Position>
```

*7.5.3.4.1.1. <*Value*> : Position value*

The Value tag defines the value of the position data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For a fixed position value, the Fixed tag must contain a character string of three real numbers, defining the values for the X, Y and Z components of the position vector.

The unit for a position value depends on the context: kilometers for a body or a satellite, meters for a satellite component or sensor.

Sample Value tag:

```
<Value>
   <Fixed Data="6755.24486 1224.32145 -3652.32185"/>
</Value>
```

7.5.3.4.2. <Orientation> : Orientation of an object

The Orientation tag defines the orientation of an object.

This tag has no attributes.

It must contain one of the available orientation type tags described below: Quaternion, EulerAngle, AxisAngle, Direction, AltAzCoordinate.

Refer to the Orientation of objects in VTS chapter for more information.

Sample Orientation tag:

```
<Orientation>
```

7.5.3.4.2.1. <Quaternion> : Orientation as a quaternion

The Quaternion tag defines the orientation of an object as a quaternion.

This tag has no attributes.

It must contain a Value tag.

Refer to the Orientation of objects in VTS chapter for more information.

Sample Quaternion tag:

```
<Quaternion>
```

<*Value*> : Quaternion value

The Value tag defines the value of the quaternion data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For a fixed quaternion value, the Fixed tag must contain a character string of four real numbers, defining the four components of the quaternion.

By convention, the first component must be the real part of the quaternion.

Sample Value tag:

```
<Value>
   <Fixed Data="1 0 0 0"/>
</Value>
```

7.5.3.4.2.2. <EulerAngle> : Orientation as Euler angles

The EulerAngles tag defines the orientation of an object as three Euler angles.

This tag has no attributes.

It must contain a Value tag.

Refer to the Orientation of objects in VTS chapter for more information.

Sample EulerAngle tag:

```
<EulerAngle>
```

<*Value*> : Euler angles values

The Value tag defines the value of the Euler angles data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For fixed Euler angles values, the Fixed tag must contain a character string of three real numbers, defining the three rotations around the Z, X' and Z'' axes.

The angles are expressed in degrees.

Sample Value tag:

```
<Value>
   <Fixed Data="90 0 45.123"/>
</Value>
```

7.5.3.4.2.3. <AxisAngle> : Orientation as an axis and angle

The AxisAngle tag defines the orientation of an object as an angle of rotation around an axis.

This tag has no attributes.

It must contain both the Axis and Angle tags.

Refer to the Orientation of objects in VTS chapter for more information.

Sample AxisAngle tag:

```
<AxisAngle>
```

<Axis> : Rotation axis

The Axis tag defines the axis of rotation for an axis and angle orientation.

This tag has no attributes.

It must contain a Value tag.

Sample Axis tag:

```
<Axis>
```

<*Value*> : Rotation axis value

The Value tag defines the value of the rotation axis data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For a fixed rotation axis value, the Fixed tag must contain a character string of three real numbers, defining the three X, Y and Z components of the rotation axis.

Sample Value tag:

```
<Value>
   <Fixed Data="0 0.707 0.707"/>
</Value>
```

<Angle> : Rotation angle

The Angle tag defines the angle of rotation for an axis and angle orientation.

This tag has no attributes.

It must contain a Value tag.

Sample Angle tag:

```
<Angle>
```

<*Value*> : Rotation angle value

The Value tag defines the value of the rotation angle data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For a fixed rotation angle value, the Fixed tag must contain a character string of one real number, defining the rotation angle.

The rotation angle is expressed in degrees.

Sample Value tag:

```
<Value>
   <Fixed Data="45.00"/>
</Value>
```

7.5.3.4.2.4. <Direction> : Orientation as a direction

The Direction tag defines the orientation of an object along a direction vector.

This tag has no attributes.

It must contain a Value tag.

Refer to the Orientation of objects in VTS chapter for more information.

Sample Direction tag:

```
<Direction>
```

<*Value*> : Direction vector value

The Value tag defines the value of the direction data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For a fixed direction value, the Fixed tag must contain a character string of three real number, defining the X, Y and Z components of the direction vector.

Sample Value tag:

```
<Value>
   <Fixed Data="0.0 -1.0 0.0"/>
</Value>
```

7.5.3.4.2.5. <AltAzCoordinate> : Orientation as an azimuth and elevation

The AltAzCoordinate tag defines the orientation of an object as an azimuth and elevation.

This tag has no attributes.

It must contain a Value tag.

Refer to the Orientation of objects in VTS chapter for more information.

Sample AltAzCoordinate tag:

```
<AltAzCoordinate>
```

<*Value*> : Azimuth and elevation values

The Value tag defines the values for the azimuth and elevation data.

Refer to the documentation of the <*Value*> generic tag in the <*Value*> : Data value section for further information.

For a fixed direction value, the Fixed tag must contain a character string of two real number, defining the azimuth and elevation.

Both the azimuth and elevation angles are expressed in degrees.

Sample Value tag:

```
<Value>
   <Fixed Data="-135.6 24.3"/>
</Value>
```

7.5.3.5. <*Events*> : Events

The Events tag defines mission events (if attached to an entity) and event decorations (project-wide or entity-specific).

This tag has no attributes.

Refer to the Mission events in VTS chapter for more information.

Sample Events tag:

```
<Events>
```

7.5.3.5.1. <*File*> : CIC/CCSDS events file

The File tag defines a CIC/CCSDS events file attached to an entity. Refer to the CIC/CCSDS event files section of the Mission events in VTS chapter for more information.

Refer to the documentation of the <*File*> generic tag in the <*File*> : CIC/CCSDS file section for further information.

7.5.3.5.2. <Decoration> : Event type decoration

The Decoration tag contains a list of decoration layer tags defining the decoration for an event type, used by client applications to display events of this type.

Attribute	Description	Format (unit)	See also
Type	Event type full name	Character string	Event type in the Mission events in VTS chapter

The Decoration tag may contain any number of Shape or Icon decoration layer tags. They should be ordered from back to front, meaning that the last layer tag in the list will be the topmost layer of the decoration.

Refer to the Configuring event types section of the VTS configuration utility user manual chapter for more information on the contents of this tag.

Sample Decoration tag:

```
<Decoration Type="ECLIPSE/DAY">
```

7.5.3.5.2.1. <Shape/> : Shape event decoration layer

The Shape tag defines a geometrical shape layer for the decoration of an event type.

Attribute	Description	Format (unit)	See also
Shape	Shape name	Character string (*Circle*, *Circular target*, *Cross*, *Diamond*, *Square*)	Configuring event types in the VTS configuration utility user manual

Attribute	Description	Format (unit)	See also
Color	Color of the shape	Three real numbers for (red, green, blue) in the [0,1] range (no unit)	
Fill	Whether or not to fill the shape	Boolean	

Sample Shape tag:

```
<Shape Shape="Circle" Color="1 1 0" Fill="1"/>
```

7.5.3.5.2.2. <Icon> : Icon event decoration layer

The Icon tag defines an icon layer for the decoration of an event type.

Attribute	Description	Format (unit)	See also
Type	Type of the icon	Character string (*Entity* or *Custom*)	Configuring event types in the VTS configuration utility user manual

If the type of the icon layer is Entity, the entity icon (i.e., its 2D symbol file) is used for the icon layer and the tag may not contain another tag. If the type of the icon layer is Custom, an icon file must be provided for the icon layer through the File tag.

Sample Icon tag:

```
<Icon Type="Entity">
```

<File/> : Icon file for an icon event decoration layer

The File tag defines a custom icon file for an icon decoration layer.

Attribute	Description	Format (unit)	See also
Name	Name of the icon file	Character string	

The File tag may only be used for Custom icon decoration layers.

The name of the icon file must be relative to the project folder.

Sample File tag:

```
<File Name="Models/sun.png">
```

7.5.3.6. <*Value*> : Data value

The Value tag defines the values for some project data.

This tag has no attributes.

It must contain one of the tags described below, according to the data source: Fixed, File or Stream.

Refer to the Data sources in VTS chapter for more information.

Sample Value tag:

```
<Value>
```

7.5.3.6.1. <Fixed/> : Fixed value

The Fixed tag defines a fixed value for some data. The fixed value is stored as an attribute to the Fixed tag.

Attribute	Description	Format (unit)	See also
Data	Fixed value for the data	Character string for the fixed value	Data sources in VTS

The actual format and unit of the Data attribute depend on the context in which the parent Value tag is used.

Sample Fixed tag:

```
<Fixed Data="1 0 0 0"/>
```

7.5.3.6.2. <*File*> : Sampled values file

The File tag defines a CIC/CCSDS file containing sampled values for some data.

Refer to the documentation of the <*File*> generic tag in the <*File*> : CIC/CCSDS file section for further information.

The actual format and unit of the file data attribute depend on the context in which the parent Value tag is used.

*7.5.3.6.2.1. <*ColorFile*> : Color file for a sampled data file*

The ColorFile tag defines a CIC/CCSDS file to provide the color associated with some sampled data.

Refer to the documentation of the <*ColorFile*> generic tag in the <*ColorFile*> : CIC/CCSDS color file section for further information.

This tag is optional.

7.5.3.6.3. <Stream/> : Streamed values

The Stream tag defines a visualization stream for the values of some data.

Attribute	Description	Format (unit)	See also
Id	Stream identifier	Character string	Data sources in VTS
Mode	Stream mode	Character string (*INTERPOL* or *DIRECT*)	Synchronization protocol for VTS clients

The actual format and unit of the streamed values depend on the context in which the parent Value tag is used.

Sample Stream tag:

```
<Stream Id="sat1" Mode="INTERPOL"/>
```

7.5.3.7. <*File*> : CIC/CCSDS file

The File tag defines a CIC/CCSDS file.

Attribute	Description	Format (unit)	See also
Name	Name of the CIC/CCSDS file	Character string	Data sources in VTS CIC/CCSDS data files in VTS

The file path must be relative to the project folder.

If this tag is used to define a sampled file for a Value tag, it may contain an optional ColorFile tag.

Sample File tag:

```
<File Name="Data/ROSETTA_OEM_POSITION.TXT"/>
```

7.5.3.8. <*ColorFile*> : CIC/CCSDS color file

The ColorFile tag defines a CIC/CCSDS color file which describes the evolution of a color with time.

Attribute	Description	Format (unit)	See also
Name	Name of the color file	Character string	

The file path must be relative to the project folder.

Sample ColorFile tag:

```
<ColorFile Name="Data/CUBESAT_MODE.TXT"/>
```

8. ABOUT

8.1. PROGRAMMING LANGUAGES IN VTS

Due to the nature of VTS, which deals with synchronization of client applications from various origins, several programming languages are used throughout the VTS toolkit. The following table lists these languages:

Language	Usage in the VTS toolkit
C++/Qt	• Main language of VTS. Used for GUIs of the Broker, the VTS configuration utility and 2DWin. • C++ is also the main language of the Celestia client application.
C	• CIC/CCSDS file I/O library.
Tcl/Tk	• PrestoPlot client application. • VTS cross-platform toolkit launcher.
Lua	• Interface scripting language in Celestia. Used by the VTS plugin for Celestia.
Java	• Language of the Orekit library, used by the orbit propagation tool.
Bash	• Compilation and deployment scripts. • Some basic client application launchers.
DOS Batch	• Some basic client application launchers.